IoData types and Processes

Technical Report PRG-TR-01-00

Christie Bolton

March 2000

Acknowledgements

This document is a slightly adapted version of the report I submitted as part of my transfer from probationer to DPhil status. As such, the work was done under the supervision of my supervisor, Jim Davies. I am greatly endebted to him for all his help and encouragement.

In addition I would like to thank Jim Woodcock for always being there to bounce ideas off, and Andy Galloway for his helpful comments about the Postie example in my transfer report.

Contents

Chapter 1

Introduction

1.1 Background

In the specification and development of large complex systems we may wish to construct a variety of models each of which addresses a different aspect of the design. Where two or more models are proposed for one component we may wish to compare these descriptions (and indeed refinements of them) in order to verify that they are consistent.

If the models are are written in the same specification language then this comparison is simply a question of refinement. If, however, they are written in different languages then some translation is required; that translation must preserve refinement.

1.2 Abstract data types and their process equivalents

In this document we explore the relationship between state-based descriptions written in Z [Spi92] and process-based descriptions written in CSP [Hoa85].

In particular, in Chapter 2 we recall the work presented in [BDW99]. We describe the translation from *abstract data types* described in Z to their process equivalents described in CSP. The *environment* of the abstract data type describes the context in which it is to be considered. The environment may be used to give the abstract data type an implicit notion of input an output by serving as a source for initial inputs and a target for final outputs.

We present a simple set of simulation rules and prove that they are both sound and complete. In addition we prove that refinement is preserved in the translation from abstract data types to their process equivalents.

1.3 ioData types and their process equivalents

In Chapter 3 we build on the work described in [BDW99] and recalled in Chapter 2. We define both semantically and syntactically *ioData types* – that is data types with an *explicit* notion of input and output – and give the translation to the corresponding ioProcesses. We prove results similar to those already established for our abstract data types: we identify a set of simulation rules for our ioData types and prove in Section 3.6 that they are sound. We conclude the document by proving in Section 3.7 the desired and expected result, that refinement of ioData types is equivalent to refinement of their corresponding ioProcesses.

1.4 Relational semantics

The *syntax* of abstract data types and ioData types tells us nothing about the result of attempting to perform an operation outside its domain of definition. It is how we choose to define our *relational semantics* which determines the behaviour of the model and hence the translation to the corresponding CSP process.

The theory of refinement introduced in [WD96] is based on a relational semantics in which operations are always available. The result of performing an operation outside its domain of definition is left undefined, and could be described by *Chaos* in the process algebra. This model has been termed a *non-blocking semantics*.

The relational semantics, and the matching translation, described in this document — both for our abstract data types and our ioData types — takes a different interpretation: operations are available only within their domain of definition; this is known as a *blocking semantics*. This is suitable for the specification of systems in which the domain of each operation has already been decided.

Chapter 2

Abstract data types and their process equivalents

In this chapter we give semantic and syntactic definitions for abstract data types and their process equivalents. We describe what it means for one process to be refined by another, for an abstract data type to be semantically refined[1] by another and for an abstract data type to be syntactically refined[1] by another.

We present theorems which state that refinement and simulation of abstract data types is equivalent to refinement of their process equivalents and that one abstract data type is refined by a second precisely when the first simulates the second.

2.1 Abstract data types

For the purposes of this document, an abstract data type is as defined below. It combines a notion of state with a collection of named operations modelled as relations. In particular, two notable operations mark *initialisation* and *finalisation*. Initialisation and finalisation are total relations, that is they are fully defined in all circumstances. All other operations on the data type may be partial.

[1]For the rest of the report, when talking about abstract data types, we will use the terms "refined by" and "simulates" to mean "semantically refined by" and "syntactically refined by" respectively.

Notation

An abstract data type is described in terms of three generic parameters:

$$[Env, Name, Local]$$

and is modelled as a Cartesian product of the type

$$\begin{aligned}
&data[Env, Name, Local] == \\
&\quad \mathbb{P}\, Local \times \\
&\qquad Env \leftrightarrow Local \times \\
&\qquad\quad Name \mathrel{+\!\!\!\rightarrow} (Local \leftrightarrow Local) \times \\
&\qquad\qquad Local \leftrightarrow Env
\end{aligned}$$

The first generic parameter, Env, represents the environment of the data type[2] and the second, $Name$, represents the set of all possible names of operations. The third generic parameter, $Local$, represents the set of all local or internal states and is hidden from the environment.

We define the following four projection functions on the data type which return the four components of the state.

$$\begin{array}{|l}
\hline
[Env, Name, Local] \\
state : data[Env, Name, Local] \rightarrow \mathbb{P}\, Local \\
init : data[Env, Name, Local] \rightarrow (Env \leftrightarrow Local) \\
ops : data[Env, Name, Local] \rightarrow Name \mathrel{+\!\!\!\rightarrow} (Local \leftrightarrow Local) \\
final : data[Env, Name, Local] \rightarrow (Local \leftrightarrow Env) \\
\hline
\forall A : data[Env, Name, Local] \bullet \\
\quad \mathrm{dom}(init\ A) = Env \ \wedge\ \mathrm{dom}(final\ A) = state\ A \ \wedge \\
\quad \mathrm{ran}(init\ A) \subseteq state\ A \ \wedge \\
\quad \forall n : \mathrm{dom}\, ops\ A \bullet \\
\qquad \mathrm{dom}(ops\ A\ n) \cup \mathrm{ran}(ops\ A\ n) \subseteq state\ A \\
\hline
\end{array}$$

The constraints ensure that initialisation and finalisation are total operations and that all other operations on the data type are relations on the state space.

[2] In Chapter 3 we will see that the idea of an environment is a convenient fiction representing the context in which the data type may be used.

Characterising the data type

The operations on the data type may (apart from initialisation and finalisation) be partial operations, and what characterises a data type is how we choose to model what happens when an operation is called outside its pre-conditions.

The model we describe here is one in which operations are blocked whenever they are called outside their domain of definition.[3]

Totalisation

We have observed that what characterises an abstract data type is how we model what happens when we call an operation outside its pre-conditions, and that in this instance we want to reflect a model in which operations thus called are blocked.

We achieve this effect by totalising the relations which describe the operations on the data type in the following way: we augment the state with the unique element $\perp$ (pronounced "bottom") denoting undefinedness and map all elements outside the domain of definition (including $\perp$ itself) onto $\perp$:

> Given an operation op on a set X, $(op : X \leftrightarrow X)$, the augmented set, written $X_\perp$, is given by:
>
> $$X_\perp = X \cup \{\perp\}$$
>
> and the totalised version of the relation, written $op^\top$, is given by:
>
> $$op^\top = op \cup \{x : X_\perp \mid x \notin \operatorname{dom} op \bullet x \mapsto \perp\}$$
>
> Every element outside the domain is mapped onto $\perp$.

Consider the simple relation $\{a \mapsto a, b \mapsto a\}$ on the set $\{a, b, c\}$ as shown in Figure 2.1. The augmented set is $\{a, b, c, \perp\}$ and the totalised version of the relation is $\{a \mapsto a, a \mapsto b, c \mapsto \perp, \perp \mapsto \perp\}$.

[3] Note that this is distinct from the model described in [WD96] in which operations are always available but if called outside their domain of definition the results may be undefined.

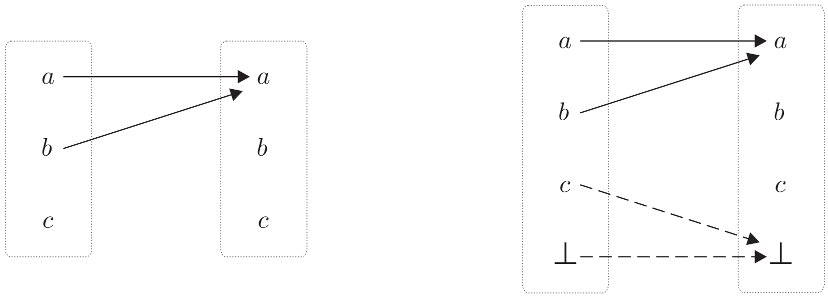

Figure 2.1: The totalisation of $\{a \mapsto a, a \mapsto b\}$.

Relational semantics

We give a *relational semantics* to our data types, describing the visible effect on the environment of every sequence of named operations as a relation on Env.

In order to define this formally we must first introduce the function $run^{\top}$ which maps each data type onto the function which takes a *program* or sequence of named operations on the data type and returns a relation on the augmented local state space. That relation is obtained by composing the totalised versions of the named operations.

We define the function $run^{\top}$ recursively[4]. For any given data type, A, the effect of the empty sequence of named operations is just the identity relation on the augmented local state space. The effect of any non-empty sequence $ns \frown \langle n \rangle$ of named operations on data type A is the result of the relational composition ($\fatsemi$) of the effect of program ns composed with the totalised version of the relation representing the named operation n.

$$
\begin{array}{|l}
[Env, Name, Local] \\
\hline
run^{\top} : data[Env, Name, Local] \rightarrow \\
\qquad\qquad \operatorname{seq} Name \mathrel{\rightarrow\mkern-14mu+} (Local_{\perp} \leftrightarrow Local_{\perp}) \\
\hline
\forall A : data[Env, Name, Local] \bullet \\
\qquad \forall n : \operatorname{dom} ops\ A;\ ns : \operatorname{seq}(\operatorname{dom} ops\ A) \bullet \\
\qquad\qquad run^{\top} A\ \ \langle\rangle = \operatorname{id}[Local_{\perp}] \ \wedge \\
\qquad\qquad run^{\top} A\ (ns \frown \langle n \rangle) = run^{\top} A\ ns \fatsemi (ops\ A\ n)^{\top} \\
\hline
\end{array}
$$

[4]Note that we have overloaded the operator $^{\top}$ in this instance. We are not totalising the relation $(run\ A\ p)$ but are totalising each operation before composing them.

We are now equipped to formally define the relational semantics of our abstract data types. For any given abstract data type, the semantics is a function from sequences of names of operations on the data type to the relation on the augmented state space obtained from the relational composition of the totalised versions of those operations preceded by initialisation and succeeded by finalisation.

$$
\begin{array}{l}
[Env, Name, Local] \\
\hline
semantics : data[Env, Name, Local] \to \\
\qquad\qquad \mathrm{seq}\, Name \rightarrowtail (Env_\perp \leftrightarrow Env_\perp) \\
\hline
\forall A : data[Env, Name, Local] \bullet \\
\quad \forall p\, \mathrm{seq}(\mathrm{dom}\, ops\, A) \bullet \\
\qquad semantics\, A\, p = \\
\qquad\quad \{e, e' : Env_\perp \mid \\
\qquad\qquad e \mapsto e' \in (init\, A)^\top \mathbin{;} run^\top\, A\, p \mathbin{;} (final\, A)^\top\}
\end{array}
$$

2.2 Corresponding processes

The processes which we describe in this section are as defined in [Hoa85] and [Ros98]. Processes are defined in terms of *events*, or synchronous atomic communications between a process and its environment.

In this section we define the class of CSP process which correspond to our abstract data types, but first we must introduce some of the necessary notation.

Notation

In order to define the process which corresponds to our abstract data types we need only introduce a small subsection of the process language.

Two pre-defined processes written into the language are *Skip* and *Stop*, representing the successful termination of a process and deadlock respectively.

The most basic operator is the prefix operator $\to$. Given an event a and a process P, the process

$$a \to P$$

is willing to perform event a and will subsequently behave like process P.

Processes may have parameters recording part of the state. If this is the case then *guards* expressed as Booleans on those parameters may determine whether or not part of a process is available. For instance, in the process below, if $B(s)$ is a Boolean on parameter s then process $P(s)$ behaves like process Q whenever $B(s)$ is true and behaves like the process *Stop*, representing deadlock, whenever $B(s)$ is false.

$$P(s) \quad = \quad B(s) \ \& \ Q$$

There are two types of choice operators denoting internal choice and external choice. An internal choice is resolved between the processes involved without reference to the environment whereas an external choice may be influenced by the environment and is resolved by the first event to occur. These two operators each come in two forms, the binary operator and the choice over an indexed set of processes.

Given processes P and Q and a set of processes $R(i)$ indexed over some set I, the following processes are as described:

$P \sqcap Q$	an internal choice between P and Q;
$P \Box Q$	an external choice between P and Q;
$\sqcap i : I \bullet R(i)$	an internal choice over the indexed set of processes;
$\Box i : I \bullet R(i)$	an external choice over the indexed set of processes.

Families of events, or *channels*, may be constructed using ".", the dot operator. These channels may be used to pass values between components. In the process P below we may externally choose some natural number n; the process will perform the event $\mathrm{in}.n$ and then behave like process Q with parameter n.

$$P \quad = \quad \Box n : \mathbb{N} \bullet \mathrm{in}.n \rightarrow Q(n)$$

Process equivalent

In this section we provide a *template* for the process equivalent of an abstract data type.

The behaviour of $\mathsf{process}(A)$, the process equivalent of abstract data type A, must reflect the behaviour of A:

- the internal state space must be hidden from the environment;

- operations called outside their pre-conditions must be blocked;
- the initial input from the environment, which named operations to perform and when to finalise are deterministically chosen;
- the states you end up in after initialisation, each operation and finalisation are non-deterministically chosen.

The process equivalent of abstract data type A may be expressed in the following format:

$$
\begin{array}{l}
\mathsf{process}\,(A) \;=\\
\quad \Box\, e : \mathit{Env} \bullet\\
\quad\quad \mathrm{init}.e \rightarrow \sqcap\, a' : \mathit{state}\ A \mid e \mapsto a' \in (\mathit{init}\ A) \bullet P(a')
\end{array}
$$

where

$$
\begin{array}{rcl}
P(a) & = & \Box\, n : \mathrm{dom}\,\mathit{ops}\ A \mid a \in \mathrm{dom}(\mathit{ops}\ A\ n) \bullet\\
& & \quad\quad \mathrm{op}.n \rightarrow\\
& & \quad\quad\quad\quad \sqcap\, a' : A \mid a \mapsto a' \in (\mathit{ops}\ A\ n) \bullet P(a')\\
& & \Box\\
& & \sqcap\, e' : \mathit{Env} \mid a \mapsto e' \in (\mathit{final}\ A) \bullet \mathrm{final}.e' \rightarrow \mathit{STOP}
\end{array}
$$

We observe that there are three channels of events; init and final which take an argument of type Env and op which takes an argument of type Name.

We may externally choose any value e of type Env (this corresponds to the initialisation operation of our abstract data type being total). The process then communicates the event $\mathrm{init}.e$ and subsequently behaves as the process $P(a')$, where the parameter a' is some value from the local state space of the data type which lies in the relational image of e under the operation $\mathit{init}\ A$. We note that the value of a' corresponds to the local state of the abstract data type; it is hidden from the environment but recorded as the parameter of the process.

In the process $P(a)$ we may externally choose to perform a named operation (under various constraints) or may choose to finalise. The choice of named operations is restricted to those operations on the data type A such that a, the parameter of the process, lies in the domain of the operation. This corresponds to the blocking of operations called outside their pre-conditions in our abstract data type. If we externally choose named operation n then the process will communicate the event $\mathrm{op}.n$ and will subsequently behave as the process $P(a')$, where a' is an internally chosen value from the relational image of a under the operation $ops\ A\ n$.

We may externally choose to finalise at any time after the init event has been communicated. This corresponds to the finalisation operation on our abstract data type being total. The process may internally choose any value e' of type Env which lies in the relational image of the parameter of the process under the operation $final\ A$ and will communicate the event final.e' and then $STOP$.

2.3 Refinement of processes

The semantic model we apply to the processes corresponding to our abstract data types is the failures model defined in [Ros98].

In this section we describe what it means under this model for one process to be refined by another. The failures of a process are expressed in terms of its possible traces and refusals sets and so first we introduce those terms.

Traces A *trace* is a history of events that a process has performed. It is expressed as a finite sequence of events in which the process has engaged up to some particular point in time with the head of the sequence being the first event that was communicated.

Refusals Whilst the set of all possible traces of a process carries a great deal of information, it does not record the availability of events. Consider processes $P1$ and $P2$ below:

$$P1 = \begin{array}{l} a \rightarrow STOP \\ \Box \\ b \rightarrow STOP \end{array} \qquad\qquad P2 = \begin{array}{l} a \rightarrow STOP \\ \sqcap \\ b \rightarrow STOP \end{array}$$

The set of traces is the same for both processes, that is

$$\{\ \langle\rangle\ , \langle a \rangle, \langle b \rangle\ \}$$

although their behaviour is different. Process $P1$ is initially willing to communicate either event a or event b before stopping whereas process $P2$ may refuse either event.

Semantically we preserve this information about availability of events in *refusals* sets. A refusal is a set of events from which a process may refuse to communicate anything no matter how long it is offered. We observe that process $P1$ may initially refuse any of the following sets of events:

- A set of events which contain neither a nor b;
- A set of events which contains a but not b;
- A set of events which contains b but not a.

It cannot, however, refuse a set which contains both events a and b.

Failures The *failures* set, $\mathit{failures}(P)$, of a process P is the set of all possible pairs, (s, A), such that the first element of the pair is a possible trace of the process and the second element of the pair is a refusal set after that trace, that is, a set of events from which the process may subsequently refuse to accept any event after it has performed the given trace.

Traces, refusals and failures of the processes corresponding to our abstract data types

For clarity we will consider only the traces, refusals and failures of the process corresponding to an abstract data type A with a single operation op.

Traces The traces of such a process can be divided into three distinct categories: the empty trace, an init event followed by any number of op events, and any number of op events preceded by an init event and succeeded by a final event. These disjoint sets of traces are described more formally below.

1. $\{\langle\rangle\}$
2. $\{n : \mathbb{N};\ e : \mathit{Env} \bullet \langle \mathrm{init}.e \rangle \frown \langle \mathrm{op} \rangle^n\}$
3. $\{n : \mathbb{N};\ e, e' : \mathit{Env} \bullet \langle \mathrm{init}.e \rangle \frown \langle \mathrm{op} \rangle^n \frown \langle \mathrm{final}.e' \rangle\}$

Refusals Initialisation on our data type is total and hence our CSP process is constructed so that initially it can only refuse sets of events containing no init.e event.

$$\mathit{refusals}\,(\,\mathsf{process}\,(A)\;/\;\langle\rangle\,)\;=\;\{\,X : \mathbb{P}\,\Sigma \mid \forall\, e : \mathit{Env} \bullet \mathrm{init}.e \notin X\,\}$$

where Σ is the set of all possible events.

After an init.e event and n op events, our process may refuse to perform another op event precisely when there is some element a from $\mathit{state}\ A$ which lies in the relational image $\mathit{init}\ A \mathbin{⨾} \mathit{run}\ A\,\langle \mathit{op}\rangle^n (\!|\ \{e\}\ |\!)$ but does not lie in the domain of $\mathit{ops}\ A\ \mathit{op}$.

In addition, after an initialisation event init.e and n op events, our process may refuse to perform a finalisation event final.e' precisely when there is some element a in the relational image $\mathit{init}\ A \mathbin{⨾} \mathit{run}\ A\,\langle \mathit{op}\rangle^n (\!|\ \{e\}\ |\!)$ such that $a \mapsto e'$ is not in the relation $\mathit{final}\ A$.

Hence the refusals set after an "unfinished" trace $\langle \mathrm{init}.e\rangle \frown \langle \mathit{op}\rangle^n$, is given by

$$\begin{array}{l}\mathit{refusals}\,(\,\mathsf{process}\,(A)\;/\;\langle \mathrm{init}.e\rangle \frown \langle \mathit{op}\rangle^n\,)\\ \quad = \;\{X : \mathbb{P}\,\Sigma \;\mid\; \exists\, a : \mathit{state}\ A \bullet\\ \qquad\qquad e \mapsto a \in \mathit{init}\ A \mathbin{⨾} \mathit{run}\ A\,\langle \mathit{op}\rangle^n \;\wedge\\ \qquad\qquad \mathrm{op} \in X \Rightarrow a \notin \mathrm{dom}\,\mathit{ops}\ A\ \mathit{op} \;\wedge\\ \qquad\qquad \forall\, e' : \mathit{Env} \bullet \mathrm{final}.e' \in X \Rightarrow a \mapsto e' \notin \mathit{final}\ A\,\}\end{array}$$

After an initialisation event, any number of op events and a finalisation event, our process may refuse any set of events. The refusals set after such a trace is given by

$$\mathit{refusals}\,(\,\mathsf{process}\,(A)\;/\;\langle \mathrm{init}.e\rangle \frown \langle \mathrm{op}\rangle^n \frown \langle \mathrm{final}.e'\rangle\,)\;=\;\{\,Y : \mathbb{P}\,\Sigma\}$$

Failures Having established the refusals sets after each of our three distinct types of traces, we can simply write down the failures of our process.

$$
\begin{array}{l}
\mathit{failures}\,(\mathsf{process}\,(A)) \\
\quad = \{ X : \mathbb{P}\,\Sigma \mid (\forall\, e : \mathit{Env} \bullet \mathrm{init}.e \notin X) \bullet (\langle\rangle\,, X) \} \\
\qquad \cup \\
\qquad \{ n : \mathbb{N};\ e : \mathit{Env};\ X : \mathbb{P}\,\Sigma \mid \\
\qquad\quad \exists\, a : \mathit{state}\ A \bullet \\
\qquad\qquad\qquad e \mapsto a \in \mathit{init}\ A \,_{9}^{\circ}\, \mathit{run}\ A\,\langle \mathit{op} \rangle^{n} \ \wedge \\
\qquad\qquad\qquad \mathrm{op} \in X \Rightarrow a \notin \mathrm{dom}\, \mathit{ops}\ A\ \mathit{op} \ \wedge \\
\qquad\qquad\qquad \forall\, e' : \mathit{Env} \bullet \mathrm{final}.e' \in X \Rightarrow a \mapsto e' \notin \mathit{final}\ A \\
\qquad\qquad (\langle \mathrm{init}.e \rangle \frown \langle \mathrm{op} \rangle^{n}, X) \} \\
\qquad \cup \\
\qquad \{ n : \mathbb{N};\ e, e' : \mathit{Env};\ X : \mathbb{P}\,\Sigma \bullet \\
\qquad\quad (\langle \mathrm{init}.e \rangle \frown \langle \mathrm{op} \rangle^{n} \frown \langle \mathrm{final}.e' \rangle, X \}
\end{array}
$$

Refinement

We say that process $\mathsf{process}\,(C)$ is a failures refinement of process $\mathsf{process}\,(A)$, written $\mathsf{process}\,(A) \sqsubseteq_F \mathsf{process}\,(C)$, if every failure of $\mathsf{process}\,(C)$ is also a failure of $\mathsf{process}\,(A)$.

$$
\begin{array}{l}
\mathsf{process}\,(A) \ \sqsubseteq_F \ \mathsf{process}\,(C) \\
\quad \Leftrightarrow \\
\quad \mathit{failures}(\mathsf{process}\,(A)) \ \supseteq \ \mathit{failures}(\mathsf{process}\,(C))
\end{array}
$$

If every behaviour of process $\mathsf{process}\,(A)$ is acceptable then so too is every behaviour of process $\mathsf{process}\,(C)$. We note that the definition of *failures* ensures that all possible traces of $\mathsf{process}\,(C)$ were also possible traces of $\mathsf{process}\,(A)$.

Failures refinement of two processes may be verified using the model checker *FDR* [For95].

2.4 Refinement of abstract data types

Recall that the semantics of any given abstract data type is a function mapping any program on the data type onto the set of effects of that program, that is, a relation on the environment *Env*.

The theory behind refinement of abstract data types is very simple: data type A is refined by data type B when the semantics of B is a subset of the semantics of A – or more precisely, for any given program on the data types, the set of effects on B is a subset of the set of effects on A.

$$A \sqsubseteq B \;\Leftrightarrow\; \forall\, p : \mathrm{seq}(\mathrm{dom}\; \mathit{ops}\; B) \bullet \mathit{semantics}\; B\; p \subseteq \mathit{semantics}\; A\; p$$

In practice, however, *proving* that B is a refinement of A is not a trivial task. We require that every possible visible effect of every possible sequence of operations on abstract data type B is a possible visible effect of the same sequence of operations on abstract data type A.

2.5 Simulation of abstract data types

Rather than attempting to establish whether or not one process refines another by considering every effect of every sequence of named operations on the data types, we can simply verify whether or not a few simple *simulation rules* (which we will introduce later in this section) hold and apply the theorem that one abstract data type is refined by another precisely when the first simulates the second.

Simulation is as described in [HHS87]. The principle behind the soundness of the simulation rules is that refinement holds in a step-by-step manner. That is, all operations on the concrete data type (including initialisation and finalisation) are subsets of the corresponding operations on the more abstract data type — or at least the *total* relations representing operations on the more concrete data type are subsets of the *total* relations representing the corresponding operations on the more abstract data type.

$$(\mathit{init}\; C)^{\top} \;\subseteq\; (\mathit{init}\; A)^{\top} \tag{2.1}$$
$$(\mathit{final}\; C)^{\top} \;\subseteq\; (\mathit{final}\; A)^{\top} \tag{2.2}$$
$$\forall\, n : \mathrm{dom}\; \mathit{ops}\; C \;\bullet\; (\mathit{ops}\; C\; n)^{\top} \;\subseteq\; (\mathit{ops}\; A\; n)^{\top} \tag{2.3}$$

One further complication is that the above rules apply only to abstract data types with the same internal state space. The abstract data types in question may, however, have different internal state spaces. When this is the case, we must identify a *retrieve relation* between the two state spaces.

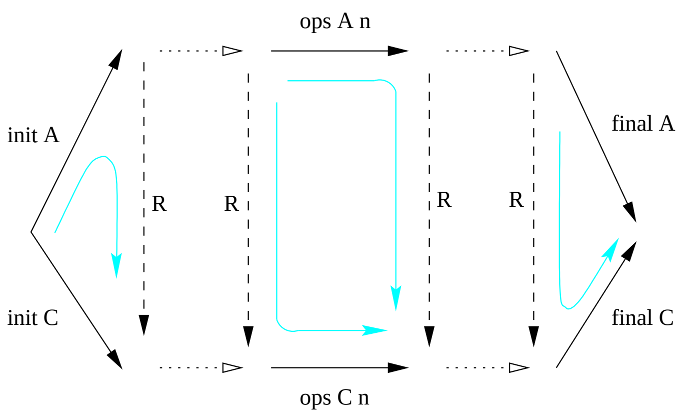

Figure 2.2: A forwards simulation.

Forwards Simulation

Given two abstract data types, A and C, we say that A forwards simulates C, written $A \preceq_{fwd} C$ precisely when it is possible to identify a relation R from *state* A to *state* C such that the following four rules[5] hold:

$$\mathit{init}\ C \subseteq (\mathit{init}\ A) \mathbin{;} R \tag{2.4}$$

$$R \mathbin{;} (\mathit{final}\ C) \subseteq \mathit{final}\ A \tag{2.5}$$

$$\forall n : \mathrm{dom}\ \mathit{ops}\ C \bullet \quad R \mathbin{;} (\mathit{ops}\ C\ n) \subseteq (\mathit{ops}\ A\ n) \mathbin{;} R \tag{2.6}$$

$$\forall n : \mathrm{dom}\ \mathit{ops}\ C \bullet \quad R \,(\!|\, \mathrm{dom}(\mathit{ops}\ A\ n) \,|\!)\, \subseteq \mathrm{dom}(\mathit{ops}\ C\ n) \tag{2.7}$$

The first three laws are as we would expect. With the necessary composition with the retrieve relation R, the initialisation relation on the more concrete data type is a subset of the initialisation relation on the more abstract data type; the finalisation relation on the more concrete data type is a subset of the finalisation relation on the more abstract data type, and the relation representing any named operation on the more concrete data type is a subset of the corresponding relation on the more abstract data type.

The fourth law, illustrated in Figure 2.3, arises from relaxing the totalised versions of the relations. Essentially the fourth law says that the image

[5]The simulation laws are given here in their *relaxed form*; that is, expressed in terms of the original relations rather than their totalised versions.

under R of any point lying in the domain of an operation on data type A must lie in the domain of the corresponding operation on C. The proof

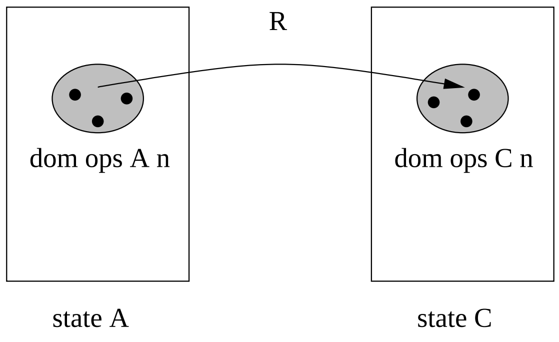

Figure 2.3: $R \llparenthesis \operatorname{dom}(ops\ A\ n) \rrparenthesis \subseteq \operatorname{dom}(ops\ C\ n)$.

that the four "relaxed" forwards simulation rules are equivalent to totalised version of the forwards simulation rules is given in [Bol98].

Backwards Simulation

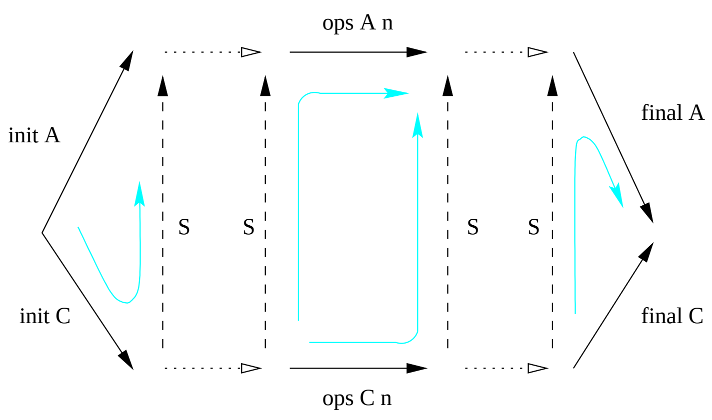

Figure 2.4: A backwards simulation

The backwards simulation rules are in essence similar. This time we need to identify a retrieve relation S from *state* C to *state* A; we say that A backwards simulates C, written $A \preceq_{bkwd} C$, precisely when the following four rules[6] hold:

$$(\mathit{init}\ C) \mathbin{;} S \ \subseteq\ \mathit{init}\ A \tag{2.8}$$

$$(\mathit{final}\ C) \ \subseteq\ S \mathbin{;} \mathit{final}\ A \tag{2.9}$$

$$\forall\, n : \mathrm{dom}\, \mathit{ops}\ A \bullet \quad (\mathit{ops}\ C\ n) \mathbin{;} S \ \subseteq\ S \mathbin{;} (\mathit{ops}\ A\ n) \tag{2.10}$$

$$\forall\, n : \mathrm{dom}\, \mathit{ops}\ A \bullet \quad (\mathit{state}\ C \setminus \mathrm{dom}\, \mathit{ops}\ C\ n) \ \subseteq\ \mathrm{dom}(S \rhd \mathrm{dom}\, \mathit{ops}\ A\ n) \tag{2.11}$$

Once again the first three laws are as we would expect and the fourth law arises from the relaxation of the totalised versions of the rules. The fourth law essentially says that applying retrieve relation S to any point outside the domain of an operation on C gives a set of points each of which lies outside the domain of the corresponding operation on A. This law is illustrated in Figure 2.5. The proof that the four "relaxed" backwards simulation rules

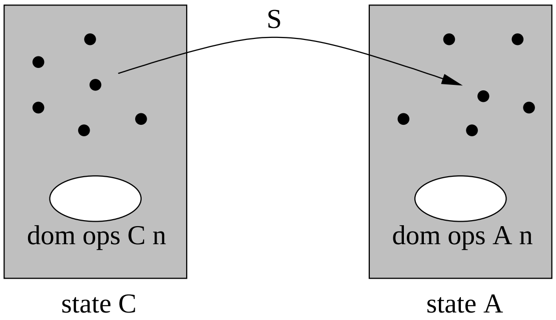

Figure 2.5: $(\mathit{state}\ C \setminus \mathrm{dom}\, \mathit{ops}\ C\ n) \ \subseteq\ \mathrm{dom}(S \rhd \mathrm{dom}\, \mathit{ops}\ A\ n)$.

are equivalent to the totalised version of the backwards simulation rules is given in [Bol98].

Simulation

Having given definitions for forwards and backwards simulation we are now equipped to define simulation. One data type, say A, simulates a second data type, say C, written $A \preceq C$, precisely when there exists a sequence of data types each of which either forwards simulates or backward simulates

[6]Once again we give the relaxed versions of the rules.

the next such that the first data type in the sequence is A and the last data type in the sequence is C.

More formally,

$$\begin{array}{l}
\forall A, C : data[Env, Name, Local] \bullet \\
\quad A \preceq C \\
\quad\quad \Leftrightarrow \\
\quad\quad (\exists ds : \mathrm{seq}\,\mathrm{data}[Env, Name, Local] \bullet \\
\quad\quad\quad (head\ ds = A \wedge last\ ds = C \ \wedge \\
\quad\quad\quad\ \forall i : 1 \mathinner{..} (\# ds - 1) \bullet \\
\quad\quad\quad\quad (ds.i \preceq_{fwd} ds.(i+1) \vee ds.i \preceq_{bkwd} ds.(i+1))))
\end{array}$$

2.6 Refinement is equivalent to simulation

We have introduced our abstract data types and described the translation to their process equivalents, and have given definitions for the refinement and simulation of abstract data types and the refinement of processes. We are now equipped to present the three main theorems that were proved in [Bol98].

The first theorem states that our simulation rules are both sound and complete: that is, one data type is refined by another precisely when the first simulates the second. This means that whenever we wish to prove that one data type is refined by another we simply need to verify that our few simple simulation rules hold.

Theorem 1
One abstract data type is refined by another precisely when the first simulates the second. □

Outline of Proof

The proof of their soundness of the simulation rules is relatively straightforward, but the proof of their completeness is of more interest.

Given two abstract data types, A and C, such that A is refined by C, we prove that A simulates C by constructing a third abstract data type,N_A, the backwards form of A. We prove that A backwards simulates N_A and

$$
\begin{tikzcd}
A \arrow[dd, "A \sqsubseteq C"'] \arrow[dr, dashed, "A \preceq_{bkwd} N_A"] & \\
& N_A \arrow[dl, dashed, "N_A \preceq_{fwd} C"] \\
C &
\end{tikzcd}
$$

Figure 2.6: We construct data type N_A which is backward simulated by A and forward simulates C

that N_A forwards simulates C. Hence by transitivity, we have shown that A simulates C.

The backwards normal form of A is constructed so that it has the same behaviour as A but delays non-determinism until finalisation. If A is of type $data[Env, Name, LocalA]$, then N_A is as defined below.

$$
\begin{array}{|l}
N_A : data[Env, Name, \mathbb{P}\, LocalA] \\
\hline
state\ N_A = (\mathbb{P}(state\ A)) \setminus \emptyset \\
init\ N_A = \{ e : Env;\ as : \mathbb{P}\, LocalA \mid as = init\ A (\! | \{e\} |\!) \} \\
\mathrm{dom}\ ops\ N_A = \mathrm{dom}\ ops\ A \\
\forall\, n : \mathrm{dom}\ ops\ N_A \bullet \\
\quad ops\ N_A\ n = \{ as, as' : \mathbb{P}\, LocalA \mid as' = ops\ A\ N (\! | as |\!) \} \\
final\ N_A = \{ as : \mathbb{P}\, LocalA;\ e : Env \mid e \in final\ A (\! | as |\!) \}
\end{array}
$$

The state space of the normal form of A is the power set of the state space of A. The initialisation operation on the normal form of A is functional and maps each element in Env onto its relational image under $init\ A$. Similarly, each operation on the normal form is functional and maps any set of elements in the state space of A onto their relational image under that operation on A. The finalisation operation on the normal form is relational and it is at this point that the non-determinism is resolved. Any set of elements in the state space of A is mapped individually onto each element in its relational image under $final\ A$.

The complete proof of the soundness and completeness of the simulation rules is given in [Bol98].

The second theorem states that refinement is preserved in the translation between data types and their equivalent processes:

Theorem 2
One abstract data type is refined by another precisely when the process corresponding to the first data type is refined by the process corresponding to the second data type. □

Once more the complete proof is given in [Bol98].

Our third theorem is really a corollary of Theorems 1 and 2, but as it sums up the essence of this chapter on abstract data types we promote it to the status of a theorem.

Theorem 3 Given two abstract data types A and C, A is refined by C precisely when A simulates C, which in turn occurs precisely when the process which corresponds to the data type A is refined by the process which corresponds to the data type C. □

The above theorem is illustrated in Figure 2.7 below.

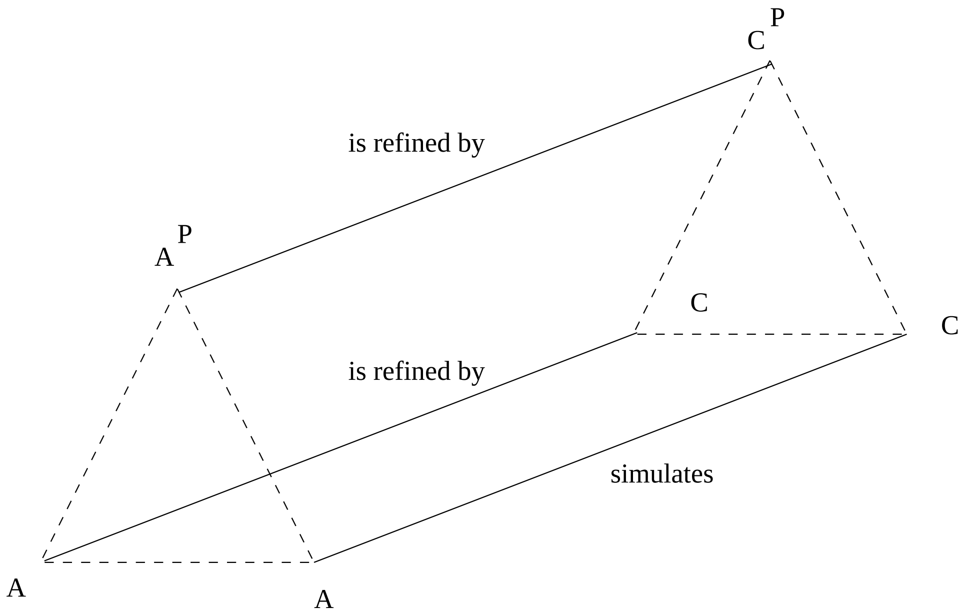

Figure 2.7: Refinement and simulation of data types and their corresponding processes.

Chapter 3

ioData types and their process equivalents

In Chapter 2 we observed that the environment of an abstract data type was a convenient fiction representing the context in which the data type may be used. In particular, the environment may serve as a source for initial input and a target for final output. In this case, inputs for all operations are *wound* into the initialisation operation and outputs from all operations are *wound* into the finalisation operation.

In this chapter we define, both syntactically and semantically, a data type (and the corresponding CSP process) which *explicitly* incorporates a notion of input and output. We then present the desired and expected theorem that refinement of ioData types is equivalent to refinement of their corresponding ioProcesses.

3.1 Syntax of ioData types

An *ioData type* is described in terms of three generic parameters:

$$[\mathit{Value}, \mathit{Name}, \mathit{Local}]$$

The first, *Value*, is the set of all possible inputs or outputs, and within a given specification may be introduced as a generic free type. The second and third parameters, *Name* and *Local* are the names of all possible operations on the data type and the set of all possible internal or local states respectively.

An ioData type is represented by a Cartesian product:

$$\begin{aligned}
&\mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] == \\
&\quad \mathbb{P}\, \mathit{Local} \times \\
&\qquad \mathit{Value} \leftrightarrow (\mathit{Local} \times \mathit{Value}) \times \\
&\qquad\quad \mathit{Name} \nrightarrow ((\mathit{Local} \times \mathit{Value}) \leftrightarrow (\mathit{Local} \times \mathit{Value})) \times \\
&\qquad\qquad (\mathit{Local} \times \mathit{Value}) \leftrightarrow \mathit{Value}
\end{aligned}$$

The following projection functions return the four parts of the state.

$$\begin{aligned}
&[\mathit{Value}, \mathit{Name}, \mathit{Local}] \\
&\mathit{ioState} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \rightarrow \mathbb{P}\, \mathit{Local} \\
&\mathit{ioInit} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \rightarrow \\
&\qquad\qquad \mathit{Value} \leftrightarrow (\mathit{Local} \times \mathit{Value}) \\
&\mathit{ioOps} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \rightarrow \\
&\qquad\qquad \mathit{Name} \nrightarrow ((\mathit{Local} \times \mathit{Value}) \leftrightarrow (\mathit{Local} \times \mathit{Value})) \\
&\mathit{ioFinal} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \rightarrow \\
&\qquad\qquad (\mathit{Local} \times \mathit{Value}) \leftrightarrow \mathit{Value} \\
\hline
&\forall\, \mathit{ioA} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \bullet \\
&\quad \operatorname{ran} \mathit{ioInit}\ \mathit{ioA} \subseteq \mathit{ioState}\ \mathit{ioA} \times \mathit{Value} \ \land \\
&\quad \operatorname{dom} \mathit{ioFinal} \subseteq \mathit{ioState}\ \mathit{ioA} \times \mathit{Value} \ \land \\
&\quad \forall\, n : \operatorname{dom} \mathit{ioOps}\ \mathit{ioA} \bullet \\
&\qquad\qquad (\operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ n \cup \operatorname{ran} \mathit{ioOps}\ \mathit{ioA}\ n) \\
&\qquad\qquad\quad \subseteq \mathit{ioState}\ \mathit{ioA} \times \mathit{Value}
\end{aligned}$$

For any given ioData type, the projection function *ioState* returns the local, or internal, state space; *ioInit* and *ioFinal* return the relations representing the initialisation and finalisation operations respectively, and *ioOps* returns a function mapping the names of operations on the ioData type onto the relation representing that operation.

The constraints ensure that all operations are relations on the Cartesian product of the state space of the ioData type and *Value* — that is, the state component lies within the state space of the ioData type. Observe that we have not insisted (as we did for our abstract data types) that the initialisation and finalisation operations are total.

In order to be able to cope with operations which have no input or no output, we introduce the "dummy" value, *null*.

$$\mathit{null} : \mathit{Value}$$

3.2 Semantics of ioData types

We give a relational semantics to our ioData types. This is obtained by considering a particular operational interpretation and may be seen as a functional relationship between *programs*, or sequences of names of operations acting on the ioData type, and the resulting visible effects of those programs on the ioData type. Once more we adopt the model in which operations called outside their pre-conditions are blocked.

When considering the relational semantics of an ioData type we wish to record the input and the output for each operation on the program. The method of presenting this information for an ioData type is not, however, entirely obvious given that the decision as to the next input is taken at each stage without knowledge of subsequent inputs and outputs.

Consider the very simple program on ioData type *ioA* illustrated in Figure 3.1.

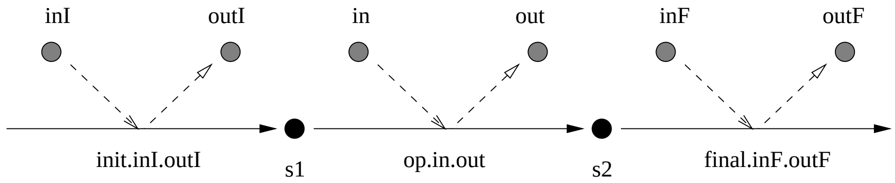

Figure 3.1: The program with the single operation op.

At initialisation we chose to input value *inI* and then received the output *outI*. We then decided to give input *in* for operation *op* and received output *out* before giving input *inF* and receiving output *outF* at finalisation.

We wish to record these inputs and outputs in our semantics which, for any given program we express as a relation between sequences of inputs and sequences of outputs. For instance, in our above example we wish the following to be true:

$$\langle inI, in, inF\rangle \mapsto \langle outI, out, outF\rangle \;\in\; ioSemantics\ ioA\ \langle op\rangle$$

In order to preserve the notion that choice of inputs is made without knowledge of subsequent inputs and outputs, this relation should be thought of as a history (after the event) of what happened.

Carrying information

We have said that the relational semantics of our ioData types are expressed in terms of sequences of input values and sequences of output values, but have defined the operations on the ioData type in terms of the local state before and after the operation, a single input value and a single output value. Therefore, in order to be able to reason about the semantics of the ioData types, we need to define functions which *carry* the necessary information about the sequences of inputs and outputs without gaining or losing any information about the operations.

First we define the state space *BatchState* which records the local state, the sequence of values still to be given as inputs, and the sequence of values which have been given as outputs.

$$
\begin{aligned}
&\mathit{BatchState}[\mathit{Local}, \mathit{Value}] == \\
&\quad (\mathit{Local} \times \mathrm{seq}\ \mathit{Value} \times \mathrm{seq}\ \mathit{Value})
\end{aligned}
$$

We also define a function *batch* which takes operations on the ioData type and returns the corresponding operation defined over *BatchState* thereby carrying forward the necessary information.

The function *batch* takes an operation on the ioData type (that is a function which takes a pair comprising a local state and an input value, and returns a pair comprising a local state and an output value) and then returns an operation defined on the state space *BatchState*.

$$
\begin{array}{l}
[\mathit{Local}, \mathit{Value}] \\
\hline
\mathit{batch} : ((\mathit{Local} \times \mathit{Value}) \leftrightarrow (\mathit{Local} \times \mathit{Value})) \rightarrow \\
\qquad (\mathit{BatchState}[\mathit{Local}, \mathit{Value}] \leftrightarrow \mathit{BatchState}[\mathit{Local}, \mathit{Value}]) \\
\hline
\forall\, \mathit{op} : (\mathit{Local} \times \mathit{Value}) \leftrightarrow (\mathit{Local} \times \mathit{Value}) \bullet \\
\qquad \mathit{batch}\ \mathit{op} = \\
\qquad\qquad \{\mathit{local}, \mathit{local}' : \mathit{Local};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value}; \\
\qquad\qquad\qquad \mathit{in}, \mathit{out} : \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad (\mathit{local}, \mathit{in}) \mapsto (\mathit{local}', \mathit{out}) \in \mathit{op} \bullet \\
\qquad\qquad\qquad\qquad\qquad (\mathit{local}, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto \\
\qquad\qquad\qquad\qquad\qquad\qquad (\mathit{local}', \mathit{ins}, \mathit{outs} \frown \langle \mathit{out} \rangle)\}
\end{array}
$$

The definition of *batch* is very much as expected: the local state and the head of the input sequence in each input triple of the returned operation are the local state and the input value of an input pair on the given operation. The output triple of the returned operation has just the tail of the input sequence – the head has been "used up" – and has the local state returned in the output pair as well as the output value returned in the output pair concatenated onto the end of its sequence of outputs.

Note that *batch op* is undefined whenever the input sequence is empty. In context this makes sense: no operation on the data type may occur without inputs and outputs.

Finally, we need to define initialisation and finalisation operations. These take in the sequence of input values and a triple of type *BatchState* respectively and return a triple of type *BatchState* and a sequence of outputs respectively.

$$
\begin{array}{l}
[\mathit{Value}, \mathit{Name}, \mathit{Local}] \\
\hline
\mathit{startBatch} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \to \\
\qquad (\mathrm{seq}\ \mathit{Value} \leftrightarrow \mathit{BatchState}[\mathit{Local}, \mathit{Value}]) \\
\hline
\forall\, \mathit{ioA} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioA} = \\
\qquad\qquad \{\, \mathit{ins} : \mathrm{seq}\ \mathit{Value};\ \mathit{local}' : \mathit{ioState}\ \mathit{ioA};\ \mathit{inI}, \mathit{outI} : \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{local}', \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (\mathit{local}', \mathit{ins}, \langle \mathit{outI} \rangle) \}
\end{array}
$$

$$
\begin{array}{l}
[\mathit{Local}, \mathit{Name}, \mathit{Value}] \\
\hline
\mathit{stopBatch} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \to \\
\qquad (\mathit{BatchState}[\mathit{Local}, \mathit{Value}] \leftrightarrow \mathrm{seq}\ \mathit{Value}) \\
\hline
\forall\, \mathit{ioA} : \mathit{ioData}[\mathit{Value}, \mathit{Name}, \mathit{Local}] \bullet \\
\qquad \mathit{stopBatch}\ \mathit{ioA} = \\
\qquad\qquad \{\, \mathit{inF}, \mathit{outF} : \mathit{Value};\ \mathit{outs} : \mathrm{seq}\ \mathit{Value};\ \mathit{local} : \mathit{ioState}\ \mathit{ioA} \mid \\
\qquad\qquad\qquad\qquad (\mathit{local}, \mathit{inF}) \mapsto \mathit{outF} \in \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad\qquad (\mathit{local}, \langle \mathit{inF} \rangle, \mathit{outs}) \mapsto \mathit{outs} \frown \langle \mathit{outF} \rangle \}
\end{array}
$$

Lifting and totalising our relations

We observed in Chapter 2 that the operations on our *abstract data types* may be partial, and that in order to propagate non-determinism until finalisation we must augment the state space with the distinguished element $\perp$ and compose the *totalised* versions of the relations. If for some program p on abstract data type A with input $e : Env$ an operation was called outside its pre-conditions then we obtained the pair $(e, \perp)$ in our semantics.

$$e \mapsto \perp \; \in \; sematics\, A\, p$$

For *ioData types*, however, it is not that straightforward. If an operation is called outside its pre-conditions when we apply program p on ioData type ioA with input sequence $\langle inI \rangle \frown ins \frown \langle inF \rangle$ then it is insufficient just to include the pair $(\langle inI \rangle \frown ins \frown \langle inF \rangle, \perp)$ in our ioSemantics.

$$\langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \perp \; \in \; ioSemantics\, ioA\, p$$

This would just tell us that at some point an operation had been called outside its pre-conditions. We would have thrown away too much information. We would not know what the actual operations that had occurred before this were; that information is tied up in the inputs and outputs.

In order to establish a one-to-one correspondence between the ioSemantics of our ioData types and the failures semantics of the corresponding ioProcesses we need to preserve the information about what outputs were given before an operation was called outside its pre-conditions. We construct the totalised[1] versions of *startBatch*, *runBatch* and *stopBatch* so that this information is preserved.

Rather than augmenting the whole state space we simply augment the component denoting the internal state space, thereby retaining all information about inputs yet to be given and outputs which have been given. When an operation is called outside its pre-conditions we map the state component onto $\perp$, use up the input at the head of the input sequence, and do not alter the output sequence. Our augmented wound state space is of the type

$$Local_{\perp} \; \times \mathrm{seq}\; Value \; \times \mathrm{seq}\; Value$$

First we consider the totalised version of *startBatch*. If the head of the initial input sequence lies in the domain of *ioInit ioA* then we just get the relation *startBatch*. If, however, the initial input sequence is $\langle inI \rangle \frown ins$

[1]It should be noted that we are to some extent abusing the term "totalising" and once more overloading the operator$^{\top}$ when referring to ioData types. The relations $startBatch^{\top}$, $stopbatch^{\top}$ and $runBatch^{\top}$ are not themselves totalised but reflect the behaviour of the totalised versions of *ioInit*, *ioFinal* and *ioOps* respectively.

for some inI which does not lie in the domain of $ioInit\ ioA$ then we map $\langle inI \rangle \frown ins$ onto the triple where the state component is the element $\bot$, the output sequence is empty and the input sequence is just ins.

$$
\begin{array}{|l}
[\,Value, Name, Local\,] \\
startBatch^{\top} : ioData[\,Value, Name, Local\,] \rightarrow \\
\qquad (\mathrm{seq}\ Value \leftrightarrow BatchState[Local_{\bot},\ Value]) \\
\hline
\forall\, ioA : ioData[\,Value, Name, Local\,] \bullet \\
\qquad startBatch^{\top}\ ioA = \\
\qquad\qquad startBatch\ ioA \\
\qquad\qquad \cup \\
\qquad\qquad \{inI : Value;\ ins : \mathrm{seq}\ Value \mid inI \notin \mathrm{dom}\ ioInit\ ioA \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (\bot, ins,\ \langle\rangle\ )\}
\end{array}
$$

Next consider the totalised versions of $batch$ and $runBatch$. There are three possible cases to consider when defining $batch^{\top}$:

- No operation has previously been called outside its pre-conditions and the pair containing the current state and the next input lies in the domain of op.
- No operation has previously been called outside its pre-conditions but the pair containing the current state and the next input does not lie in the domain of op.
- An operation has already been called outside its pre-conditions.

In the first case we simply have the relation $batch\ (ioA\ op)$.

In the second case, we want to map our input triple $(a, \langle in \rangle \frown ins, outs)$ onto the triple $(\bot, ins, outs)$. The $\bot$ in the state space indicates that an operation has been called outside its pre-conditions and no more outputs will be given. The head of the input sequence has been 'used up', and the output sequence remains the same.

In the third case, the state component is already be bottom and will be mapped onto bottom. The head of the input sequence will be 'used up' and the output sequence will not be changed.

$$
\begin{array}{l}
[Local, Value] \\
\hline
batch^{\top} : (Local \times Value) \leftrightarrow (Local \times Value) \rightarrow \\
\qquad BatchState[Local_{\perp}, Value] \leftrightarrow BatchState[Local_{\perp}, Value] \\
\hline
\forall\, op : (Local \times Value) \leftrightarrow (Local \times Value) \bullet \\
\qquad batch^{\top}\ op = \\
\qquad\qquad batch\ op \\
\qquad\qquad \cup \\
\qquad\qquad \{\, local : Local;\ in : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ op \bullet \\
\qquad\qquad\qquad\qquad (local, \langle in \rangle \frown ins, outs) \mapsto (\perp, ins, outs) \} \\
\qquad\qquad \cup \\
\qquad\qquad \{\, in : Value;\ ins, outs : \mathrm{seq}\ Value \bullet \\
\qquad\qquad\qquad (\perp, \langle in \rangle \frown ins, outs) \mapsto (\perp, ins, outs) \}
\end{array}
$$

Given this definition, $runBatch^{\top}$ is easy to define iteratively.

$$
\begin{array}{l}
[Value, Name, Local] \\
\hline
runBatch^{\top} : ioData[Value, Name, Local] \rightarrow (\mathrm{seq}\ Name \nrightarrow \\
\qquad BatchState[Local_{\perp}, Value] \leftrightarrow BatchState[Local_{\perp}, Value]) \\
\hline
\forall\, ioA : \mathrm{ioData}[Value, Name, Local] \bullet \\
\qquad runBatch^{\top}\ ioA\ \ \langle\rangle = \mathrm{id}[BatchState[Local_{\perp}, Value] \\
\qquad \wedge \\
\qquad \forall\, n : \mathrm{dom}\ ioOps\ ioA;\ ns : \mathrm{seq}(\mathrm{dom}\ ioOps\ ioA) \bullet \\
\qquad\qquad runBatch^{\top}\ ioA\ (ns \frown \langle n \rangle) = \\
\qquad\qquad\qquad runBatch^{\top}\ ioA\ ns \fatsemi (batch^{\top}\ (ioOps\ ioA\ n))
\end{array}
$$

At finalisation there are once more the three cases:

- No operation has previously been called outside its pre-conditions and the pair containing the current state and the next input lies in the domain of $ioFinal\ ioA$. In this case we just get the relation $stopBatch$.
- No operation has previously been called outside its pre-conditions but the pair containing the current state and the next input does not lie in the domain of $ioFinal\ ioA$. In this case we output the sequence of outputs given so far which will be of length one more than the length of the program.

- An operation has already been called outside its pre-conditions. In this case the length of the output sequence will be less than or equal to the length of the program (and will be the empty sequence if it was the initialisation operation which was called outside its pre-conditions).

$$
\begin{array}{l}
[Local, Name, Value] \\
\hline
stopBatch^{\top} : ioData[Value, Name, Local] \rightarrow \\
\qquad BatchState[Local_{\perp}, Value] \leftrightarrow \mathrm{seq}\ Value \\
\hline
\forall\, ioA : ioData[Value, Name, Local] \bullet \\
\qquad stopBatch^{\top}\ ioA = \\
\qquad\qquad stopBatch\ ioA \\
\qquad\qquad \cup \\
\qquad\qquad \{local : Local;\ in : Value;\ outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad\qquad (local, \langle in \rangle, outs) \mapsto outs\} \\
\qquad\qquad \cup \\
\qquad\qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\}
\end{array}
$$

Combining these we get our definition of our ioSemantics which, for any ioData type and program on that ioData type is a relation between sequences of *Values*. The length of the input sequence is precisely two more than the length of the program and the length of the output sequence is at most two more than the length of the program; its length indicates where things went wrong if they did, leading nicely into our failures model.

$$
\begin{array}{l}
[Value, Name, Local] \\
\hline
ioSemantics : ioData[Value, Name, Local] \rightarrow \\
\qquad \mathrm{seq}\ Name \mathrel{+\!\!\!\rightarrow} (\mathrm{seq}\ Value \leftrightarrow \mathrm{seq}\ Value) \\
\hline
\forall\, ioA : ioData[Value, Name, Local] \bullet \\
\qquad \forall\, p : \mathrm{seq}(\mathrm{dom}\ ioOps\ ioA) \bullet \\
\qquad\qquad ioSemantics\ ioA\ p = \\
\qquad\qquad\qquad \{ins, outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad ins \mapsto outs' \in \\
\qquad\qquad\qquad\qquad\qquad startBatch^{\top}\ ioA \mathbin{⨾} \\
\qquad\qquad\qquad\qquad\qquad\qquad runBatch^{\top}\ ioA\ p \mathbin{⨾} stopBatch^{\top}\ ioA\}
\end{array}
$$

3.3 Corresponding ioProcesses

In this section we provide, as we did for our abstract data types, a template for the ioProcesses corresponding to our ioData types. Once more the behaviour of these processes must reflect the behaviour of the data types they correspond to.

- The ioProcess must have an initialisation operation that requires an input and gives an output;
- The internal state space of the process must be hidden from the environment;
- All operations must have an input and give an output;
- Choice of the inputs as well as which named operations to perform and when to finalise must be deterministic;
- The state you end up in after each operation and the outputs given must be non-deterministic;
- Operations called outside their pre-conditions must be blocked.

Given ioData type, ioA, the corresponding CSP process $\mathsf{ioProcess}\,(ioA)$ is given by:

$$
\begin{array}{l}
\mathsf{ioProcess}\,(ioA) = \\
\quad \Box\, inI : Value \mid inI \in \mathrm{dom}\, ioInit\; ioA \bullet \\
\quad\quad \sqcap\, a' : state\; ioA;\; outI : Value \mid inI \mapsto (a', outI) \in ioInit\; ioA \bullet \\
\quad\quad\quad \mathrm{init}.inI.outI \to P(a')
\end{array}
$$

$$
\begin{array}{l}
\mathsf{where} \\
P(a) = \\
\quad \Box\, op : \mathrm{dom}\, ioOps\; ioA;\; in : Value \mid (a, in) \in \mathrm{dom}\, ioOps\; ioA\; op \bullet \\
\quad\quad \sqcap\, a' : ioState\; ioA;\; out : Value \mid \\
\quad\quad\quad\quad (a, in) \mapsto (a', out) \in ioOps\; ioA\; op \bullet \\
\quad\quad\quad \mathrm{op}.in.out \to P(a') \\
\quad \Box \\
\quad \Box\, inF : Value \mid (a, inF) \in ioFinal\; ioA \bullet \\
\quad\quad \sqcap\, outF : Value \mid (a, inF) \mapsto outF \in ioFinal\; ioA \bullet \\
\quad\quad\quad \mathrm{final}.inF.outF \to Stop
\end{array}
$$

Initially we may externally choose any value inI of type $Value$ which lies in the domain of the operation $ioInit\; ioA$. The process will then internally choose some pair $(a', outI)$ from the relational image of inI under $ioInit\ ioA$ and will communicate the event init.$inI.outI$, outputting the value $outI$ and will subsequently behave like the process P with parameter a'.

In the process $P(a)$ we may externally choose an event corresponding to one of the named operations on the ioData type or choose to finalise.

We may only choose to perform an event corresponding to a named operation op if we give an input in such that (a, in) lies in the domain of the relation $ioOps\ ioA\ op$. If this is the case then the process will communicate the event op.$in.out$ for some internally chosen value out and will subsequently behave as process P with parameter a' where a' is an internally chosen value of type $ioState\ ioA$ such that (a', out) lies in the relational image of (a, in) under $ioOps\ ioA\ op$.

We may externally choose to finalise provided we externally choose some value inF such that (a, inF) lies in the domain of the relation $ioFinal\ ioA$. If this is the case then the process will internally choose any value $outF$ which lies in the relational image of (a, inF) under $ioFinal\ ioA$ and communicate the event final.$inF.outF$ and then $STOP$.

3.4 Refinement of ioProcesses

For clarity we consider only the failures of ioData types with a single operation op but the results obtained may be extended to all ioData types.

Traces, refusals and failures of ioProcesses

Traces As with our abstract data types, the set of all possible traces of an ioProcess may be expressed as the disjoint union of three sets: the set containing the empty trace, the set of all unfinished traces and the set of all finished traces.

Given the definition of our ioProcesses we observe that for all natural numbers n, values inI and $outI$ and sequences of values ins and $outs$ of length n, the sequence[2]

$$\mathrm{init}.inI.outI \frown \langle r : 1 \,..\, n \bullet \mathrm{op}.(ins\ r).(outs\ r)\rangle$$

[2] We use the notation $\langle r : 1 \,..\, n \bullet \mathrm{op}.(ins\ r).(outs\ r)\rangle$ to denote the successive concate-

is an (unfinished) trace of $\mathsf{ioProcess}\,(ioA)$ precisely when there exists a sequence of local states, $as : \mathrm{seq}(\mathrm{dom}\, ioState\; ioA)$ of length $n+1$ such that:

$$\begin{array}{l} inI \mapsto (head\; as, ins\; 1) \;\in\; init\; ioA \\ \wedge \\ \forall\, i : 1\,..\,n \;\bullet\; (as\; i, ins\; i) \mapsto (as\; (i+1), outs\; i) \;\in\; ioOps\; ioA\; op \end{array}$$

Recalling the definitions of *startBatch* and *runBatch*, this is equivalent to saying that for all natural numbers n, values inI and $outI$ and sequences of values ins and $outs$ of length n, the sequence

$$\mathrm{init}.inI.outI \frown \langle r : 1\,..\,n \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle$$

is an (unfinished) trace of $\mathsf{ioProcess}\,(ioA)$ precisely when there exists a local state $a : ioState\; ioA$ such that

$$\begin{array}{l} \langle inI\rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI\rangle \frown outs) \\ \quad \in \\ \quad startBatch\; ioA \fatsemi runBatch\; ioA \langle \mathrm{op}\rangle^{n} \end{array}$$

The set of unfinished traces for $\mathsf{ioProcess}\,(ioA)$ is therefore given by

$$\begin{array}{l} \{\, n : \mathbb{N};\ inI, outI : Value;\ ins, outs : \mathrm{seq}\; Value \ \mid \\ \qquad (\#ins = \#outs = n \\ \qquad \wedge \\ \qquad \exists\, a : ioState\; ioA \bullet \\ \qquad\qquad \langle inI\rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI\rangle \frown outs) \\ \qquad\qquad\qquad \in \\ \qquad\qquad\qquad startBatch\; ioA \fatsemi runBatch\; ioA \langle \mathrm{op}\rangle^{n}) \\ \qquad\quad \bullet\ \langle initOp\rangle \frown \langle r : 1\,..\,n \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \,\} \end{array}$$

Following a similar argument we see that the set of finished traces for

---

nation of op events, that is

$$\begin{array}{l} \forall\, n : \mathbb{N};\ ins, outs : \mathrm{seq}\; Value \mid \#ins = \#outs = n \;\bullet \\ \quad \langle r : 1\,..\,0 \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \;=\; \langle\rangle \\ \quad \wedge \\ \quad \langle r : 1\,..\,(n+1) \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \;= \\ \qquad \langle r : 1\,..\,n \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \frown \langle \mathrm{op}.(ins\; (n+1)).(outs\; (n+1))\rangle \end{array}$$

$\mathsf{ioProcess}\,(ioA)$ must be given by

$$\begin{array}{l}\{\, n : \mathbb{N};\ \ inI, outI, outF, outF \,:\, Value;\ ins, outs : \mathrm{seq}\, Value \ \mid \\ \qquad (\#ins = \#outs = n \\ \qquad \wedge \\ \qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \\ \qquad\qquad\qquad \in \\ \qquad\qquad\qquad startBatch\ ioA \fatsemi runBatch\ ioA \langle \mathrm{op} \rangle^n \fatsemi stopBatch\ ioA\,) \\ \qquad\quad \bullet \langle initOp \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle \,\}\end{array}$$

Refusals after the empty trace At initialisation $\mathsf{ioProcess}\,(ioA)$ may refuse a set X if the following four conditions hold:

- the set contains no $\mathrm{init}.inI.outI$ events;
- for each event $\mathrm{init}.inI.outI$ in the set X, that value inI does not lie in the domain of the operation $ioInit\ ioA$;
- for each event $\mathrm{init}.inI.outI$ in the set X, such that value inI does lie in the domain of the operation $ioInit\ ioA$, the value $outI$ does not lie in the range of the relational image of inI under $ioInit\ ioA$;
- for each event $\mathrm{init}.inI.outI$ in the set X, such that value inI does lie in the domain of the operation $ioInit\ ioA$ and the value $outI$ does lie in the range of the relational image of inI under $ioInit\ ioA$, there is some other value $outI'$ such that $outI'$ also lies in the relational image of inI under $ioInit\ ioA$ but the event $\mathrm{init}.inI.outI'$ does not lie in the set X.[3]

We may express these four conditions succinctly as follows[4]:

$$\begin{array}{l}\forall X : \mathbb{P}\,\Sigma \bullet \\ \qquad X \in refusals\ (\mathsf{ioProcess}\,(ioA)) \\ \qquad\quad \Leftrightarrow \\ \qquad\quad (\forall\, inI, outI \,:\, Value \mid inI \in \mathrm{dom}\, ioInit\ ioA \bullet \\ \qquad\qquad\quad \mathrm{init}.inI.outI \in X \\ \qquad\qquad\qquad \Rightarrow \\ \qquad\qquad\qquad (\exists\, outI' : \mathrm{ran}(ioInit\ ioA (\!|\ \{inI\}\ |\!)) \ \bullet \\ \qquad\qquad\qquad\quad \mathrm{init}.inI.outI' \notin X\,)\,)\end{array}$$

[3] Recall that $\{a\} \in refusals\ ((a \rightarrow STOP) \sqcap (b \rightarrow STOP))$.

[4] Σ is the set of all possible events.

It is immediately obvious that this covers the fourth condition.

It covers the third condition because we know that $outI$ does not lie in the range of $ioInit\ ioA$ and so, since we also know that inI lies in the domain of $ioInit\ ioA$, there must be some other value $outI'$ such that $outI'$ does lie in the range of $ioInit\ ioA$; either $\mathrm{init}.inI.outI'$ is not contained in X in which case the predicate is automatically true, or it is contained in X in which case by the fourth condition there must be yet another value $outI''$ such that $outI''$ lies in the range of $ioInit\ ioA$ but the event $\mathrm{init}.inI.outI''$ is not contained in X.

The second condition is covered because no constraint is put on sets containing events $\mathrm{init}.inI.outI$ where inI is not in the domain of $ioInit\ ioA$.

The first condition is trivially covered because no constraint is put on a set which contains no init events.

The subset of the failures of ioProcess (ioA) in which the trace component is the empty trace is therefore given by

$$
\begin{array}{l}
\{X : \mathbb{P}\Sigma \mid \\
\quad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\ ioInit\ ioA \bullet \\
\qquad \mathrm{init}.inI.outI \in X \\
\qquad\quad \Rightarrow \\
\qquad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioA ⦇ \{inI\} ⦈) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad \bullet (\langle\rangle, X)\}
\end{array}
$$

Refusals after an unfinished trace

Recall that an unfinished trace is of the form $\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathrel{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r)\rangle$ for some natural number n, values inI and $outI$ and sequences of values ins and $outs$ of length n.

After such a trace, the process may refuse a set X containing the event $\mathrm{op}.in.out$ precisely when there is some Local state a that could have been reached after the given trace such that one of the following is true:

- out does not lie in the range of the relational image of $ioOps\ ioA\ op$ under (a, in);
- there is some value out' not in the range of the relational image of $ioOps\ ioA\ op$ under (a, in), such that the event $\mathrm{op}.in.out'$ is not in the set X.

Similarly, after such a trace, the process may refuse a set X containing the event final.$inF.outF$ precisely when there is some Local state a that could have been reached after the given trace such that

- $outF$ does not lie in the relational image of $ioFinal\ ioA$ under (a, inF)
- there is some value $outF'$ not in the relational image of $ioFinal\ ioA$ under (a, inF), such that the event final.$inF.outF'$ is not in the set X.

Combining these observations we see that the subset of failures of process ioProcess (ioA) in which the trace component is an unfinished traces is given by:

$$
\begin{array}{l}
\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad (\ \#ins = \#outs = n \\
\quad\ \wedge \\
\quad\ \exists\, a : Local \bullet \\
\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad startBatch\ ioA \mathbin{⨾} runBatch\ ioA \langle op \rangle^{n} \\
\qquad \wedge \\
\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\ ioOps\ \ ioA\ op \bullet \\
\qquad\qquad \mathrm{op}.in.out \in X \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \wedge \\
\qquad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad \mathrm{final}.inF.outF \in X \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \\
\qquad\qquad\qquad\qquad \mathrm{final}.inF.outF' \notin X\ )) \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}
\end{array}
$$

Refusals after finished traces

The process may refuse any set of events after a finished trace, and therefore the subset of failures containing finished traces is given by:

$$
\begin{array}{l}
\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad \#ins = \#outs = n \\
\quad \wedge \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \\
\qquad\qquad \in \\
\qquad\qquad startBatch\ ioA \mathbin{⨾} runBatch\ ioA \langle op \rangle^{n} \mathbin{⨾} stopBatch\ ioA \\
\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \\
\qquad\qquad \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\}
\end{array}
$$

The Failures of ioProcess (ioA)

We now combine all of these results to express the failures of ioProcess (ioA) as the disjoint union of its failures after the empty trace, its failures after all unfinished traces and its failures after all finished traces:

$$
\begin{array}{l}
\mathit{failures}\ (\mathsf{ioProcess}\,(ioA))\ = \\
\quad \{X : \mathbb{P}\,\Sigma \mid \\
\quad\quad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\ ioA \bullet \\
\quad\quad\quad \mathrm{init}.inI.outI \in X \\
\quad\quad\quad\quad \Rightarrow \\
\quad\quad\quad\quad \exists\, outI' : \operatorname{ran}(ioInit\ ioA(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\quad\quad\quad\quad\quad \bullet\ (\ \langle\rangle\ , X)\} \\
\quad \cup \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \operatorname{seq} Value \mid \\
\quad\quad (\ \#ins = \#outs = n \\
\quad\quad\ \wedge \\
\quad\quad\ \exists\, a : Local \bullet \\
\quad\quad\quad \langle inI\rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\quad\quad\quad\quad startBatch\ ioA \fatsemi runBatch\ ioA\langle op\rangle^{n} \\
\quad\quad\quad \wedge \\
\quad\quad\quad \forall\, in, out : Value \mid (a, in) \in \operatorname{dom} ioOps\ \ ioA\ op \bullet \\
\quad\quad\quad\quad \mathrm{op}.in.out \in X \\
\quad\quad\quad\quad\quad \Rightarrow \\
\quad\quad\quad\quad\quad \exists\, out' : \operatorname{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \\
\quad\quad\quad\quad\quad\quad \mathrm{op}.in.out' \notin X \\
\quad\quad\quad \wedge \\
\quad\quad\quad \forall\, inF, outF : Value \mid (a, inF) \in \operatorname{dom} ioFinal\ ioA \bullet \\
\quad\quad\quad\quad \mathrm{final}.inF.outF \in X \\
\quad\quad\quad\quad\quad \Rightarrow \\
\quad\quad\quad\quad\quad \exists\, outF' : \operatorname{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \\
\quad\quad\quad\quad\quad\quad \mathrm{final}.inF.outF' \notin X\ ) \\
\quad\quad\quad\quad \bullet\ (\langle \mathrm{init}.inI.outI\rangle \frown \langle r : 1\,..\,n \bullet \mathrm{op}.(ins\ r).(outs\ r)\rangle, X)\} \\
\quad \cup \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \operatorname{seq} Value \mid \\
\quad\quad \#ins = \#outs = n \\
\quad\quad \wedge \\
\quad\quad \langle inI\rangle \frown ins \frown \langle inF\rangle \mapsto \langle outI\rangle \frown outs \frown \langle outF\rangle \in \\
\quad\quad\quad\quad startBatch\ ioA \fatsemi runBatch\ ioA\langle op\rangle^{n} \fatsemi stopBatch\ ioA \\
\quad\quad\quad \bullet\ (\langle \mathrm{init}.inI.outI\rangle \frown \\
\quad\quad\quad\quad \langle r : 1\,..\,n \bullet \mathrm{op}.(ins\ r).(outs\ r)\rangle \frown \langle \mathrm{final}.inF.outF\rangle, X)\}
\end{array}
$$

Refinement

As with the processes corresponding to our abstract data types, we use the stable failures refinement model for our ioProcesses; given two ioProcesses, the first is refined by the second when the failures of the second is a subset of the failures of the first.

$$\begin{aligned}&\mathsf{ioProcess}\,(ioA) \;\sqsubseteq_F\; \mathsf{ioProcess}\,(ioC)\\ &\quad\Leftrightarrow\\ &\quad failures\,(\,\mathsf{ioProcess}\,(ioA)\,) \;\supseteq\; failures\,(\,\mathsf{ioProcess}\,(ioC)\,)\end{aligned}$$

The may be mechanically verified using the model-checker FDR.

3.5 Refinement of ioData types

We say that one ioData type is refined by another when the set of effects of a sequence of operations on the first is a subset of the set of set of effects of the same sequence of operations on the second.

$$\begin{aligned}&ioA \;\sqsubseteq\; ioC\\ &\quad\Leftrightarrow\; \forall\, p : \mathrm{seq}\,\mathrm{dom}\, ioOps\; ioC \bullet\\ &\qquad\qquad\qquad ioSemantics\; ioA \;\supseteq\; ioSemantics\; ioC\end{aligned}$$

Whilst this definition is very simple, recalling the definition of the semantics of ioData types,

$$\begin{array}{|l}
[\,Value, Name, Local\,]\\
ioSemantics : ioData[\,Value, Name, Local\,] \rightarrow\\
\qquad\qquad \mathrm{seq}\, Name \rightarrow\!\!\!\!\!+ (\mathrm{seq}\; Value \leftrightarrow \mathrm{seq}\; Value)\\
\hline
\forall\, ioA : ioData[\,Value, Name, Local\,] \bullet\\
\quad \forall\, p : \mathrm{seq}(\mathrm{dom}\; ioOps\; ioA) \bullet\\
\qquad ioSemantics\; ioA\; p =\\
\qquad\quad \{ins, outs' : \mathrm{seq}\; Value \mid\\
\qquad\qquad ins \mapsto outs' \in\\
\qquad\qquad\quad startBatch^{\top}\; ioA \,;\\
\qquad\qquad\qquad runBatch^{\top}\; ioA\; p \,;\, stopBatch^{\top}\; ioA\}
\end{array}$$

it is not easy and may be very laborious to prove *directly* that one ioData type is refined by another. However, as with abstract data types, we may prove *indirectly* that one is refined by another by verifying that one simulates the other using a simple set of simulation rules.

3.6 Simulation of ioData types

In this section we present both the totalised and relaxed versions of the forwards and backwards simulation rules for ioData types and prove that they are sound

Forwards simulation

Given two ioData types ioA and ioC of types $ioData[LocalA, Name, Value]$ and $ioData[LocalC, Name, Value]$ respectively, we say that ioA forwards simulates ioC precisely when there exists some retrieve relation r

$$r : LocalA \leftrightarrow LocalC$$

and hence some retrieve relation R

$$\begin{array}{l} R : (\, LocalA \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \leftrightarrow \\ \qquad (\, LocalC \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \\ \hline R = \{a : LocalA;\ c : LocalC;\ ins, outs : \mathrm{seq}\ Value \mid \\ \qquad a \mapsto c \in r \bullet (a, ins, outs) \mapsto (c, ins, outs)\} \end{array}$$

with lifted version R^T where

$$\begin{array}{l} R^T : (\, LocalA_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \leftrightarrow \\ \qquad (\, LocalC_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \\ \hline R^T \ = \ R \cup \mathrm{id}[\{\perp\} \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value] \end{array}$$

such that the following three rules hold:

$$startBatch^T\ ioC \ \subseteq\ startBatch^T\ ioA \ ⨾\ R^T \tag{3.1}$$

$$R^T ⨾ stopBatch^T\ ioC \ \subseteq\ stopBatch^T\ ioA \tag{3.2}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\, ioOps\ ioC \bullet \\ \qquad R^T ⨾ batch^T\ ioOps\ ioC\ n \ \subseteq\ batch^T\ ioOps\ ioA\ n ⨾ R^T \end{array} \tag{3.3}$$

The three totalised forwards simulation rules may be relaxed to the following equivalent set of six rules:

$$ioInit\ ioC \ \subseteq\ ioInit\ ioA ⨾ (r \parallel \mathrm{id}) \tag{3.4}$$

$$(r \parallel \mathrm{id}) ⨾ ioFinal\ ioC \ \subseteq\ ioFinal\ ioA \tag{3.5}$$

$$\forall\, n : \operatorname{dom} ioOps\ ioC \bullet \qquad (3.6)$$
$$(r \parallel \mathrm{id}) \,⨾\, ioOps\ ioC\ n \ \subseteq\ ioOps\ ioA\ n \,⨾\, (r \parallel \mathrm{id})$$

$$\operatorname{dom} ioInit\ ioA \ =\ \operatorname{dom} ioInit\ ioC \qquad (3.7)$$

$$(r \parallel \mathrm{id})(\!|\ \operatorname{dom} ioFinal\ ioA\ |\!) \ \subseteq\ \operatorname{dom} ioFinal\ ioC \qquad (3.8)$$

$$\forall\, n : \operatorname{dom} ioOps\ ioC \bullet \qquad (3.9)$$
$$(r \parallel \mathrm{id})(\!|\ \operatorname{dom} ioOps\ ioA\ n\ |\!) \ \subseteq\ \operatorname{dom} ioOps\ ioC\ n$$

The proof of the equivalence of the relaxed and totalised versions of the forwards simulations rules is given in Section A.1 in the Appendix.

Backwards simulation

Given two ioData types ioA and ioC of types $ioData[LocalA, Name, Value]$ and $ioData[LocalC, Name, Value]$ respectively, we say that ioA backwards simulates ioC precisely when there exists some retrieve relation s

$$s : LocalC \leftrightarrow LocalA$$

and hence some retrieve relation S

$$\begin{array}{l} S : (\,LocalC \times \operatorname{seq} Value \times \operatorname{seq} Value\,) \ \leftrightarrow \\ \qquad\quad (\,LocalA \times \operatorname{seq} Value \times \operatorname{seq} Value\,) \\ \hline S = \{a : LocalA;\ c : LocalC;\ ins, outs : \operatorname{seq} Value \mid \\ \qquad\qquad c \mapsto a \in s \bullet (c, ins, outs) \mapsto (a, ins, outs)\} \end{array}$$

with lifted version S^T where

$$\begin{array}{l} S^T : (\,LocalC_\perp \times \operatorname{seq} Value \times \operatorname{seq} Value\,) \ \leftrightarrow \\ \qquad\qquad (\,LocalA_\perp \times \operatorname{seq} Value \times \operatorname{seq} Value\,) \\ \hline S^T \ =\ S \cup \mathrm{id}[\{\perp\} \times \operatorname{seq} Value \times \operatorname{seq} Value] \end{array}$$

such that the following three rules hold:

$$startBatch^T\ ioC \,⨾\, S^T \ \subseteq\ startBatch^T\ ioA \qquad (3.10)$$

$$stopBatch^T\ ioC \ \subseteq\ S^T \,⨾\, stopBatch^T\ ioA \qquad (3.11)$$

$$\forall\, n : \operatorname{dom} ioOps\ ioC \bullet \qquad (3.12)$$
$$batch^T\ ioOps\ ioC\ n \,⨾\, S^T \ \subseteq\ S^T \,⨾\, batch^T\ ioOps\ ioA\ n$$

The three totalised backwards simulation rules may be relaxed to the following equivalent set of six rules:

$$\mathit{ioInit}\ \mathit{ioC} \mathbin{;} (s \parallel \mathrm{id}) \subseteq \mathit{ioInit}\ \mathit{ioA} \tag{3.13}$$

$$\mathit{ioFinal}\ \mathit{ioC} \subseteq (s \parallel \mathrm{id}) \mathbin{;} \mathit{ioFinal}\ \mathit{ioA} \tag{3.14}$$

$$\begin{aligned} &\forall\, n : \mathrm{dom}\, \mathit{ioOps}\ \mathit{ioC} \bullet \\ &\quad \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} (s \parallel \mathrm{id}) \subseteq (s \parallel \mathrm{id}) \mathbin{;} \mathit{ioOps}\ \mathit{ioA}\ n \end{aligned} \tag{3.15}$$

$$\mathrm{dom}\, \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\, \mathit{ioInit}\ \mathit{ioC} \tag{3.16}$$

$$\begin{aligned} &(\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\, \mathit{ioFinal}\ \mathit{ioC} \\ &\quad \subseteq \\ &\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\, \mathit{ioFinal}\ \mathit{ioA}) \end{aligned} \tag{3.17}$$

$$\begin{aligned} &\forall\, n : \mathrm{dom}\, \mathit{ioOps}\ \mathit{ioC} \bullet \\ &\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\, \mathit{ioOps}\ \mathit{ioC}\ n \\ &\qquad \subseteq \\ &\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\, \mathit{ioOps}\ \mathit{ioA}\ n) \end{aligned} \tag{3.18}$$

The proof of the equivalence of the relaxed and totalised versions of the forwards simulations rules is given in Section A.2 of the Appendix.

Simulation implies refinement

Having identified the forwards and backwards simulation rules for ioData types we now prove that these rules are sound. We prove that when one ioData type *forward simulates* a second then the first is refined by the second and that when one ioData type *backward simulates* a second then the first is refined by the second. It then follows from the transitivity definition of simulation that when one ioData type *simulates* a second then the first is refined by the second.

Theorem 4 Given two ioData types, if the first simulates the second then the first is refined by the second. □

Proof

Let ioA and ioC be of types $\mathit{ioData}[\mathit{LocalA}, \mathit{Name}, \mathit{Value}]$ and $\mathit{ioData}[\mathit{LocalC}, \mathit{Name}, \mathit{Value}]$ respectively.

$$\mathit{ioA} \preceq \mathit{ioC}$$

$$\begin{array}{ll}
\Leftrightarrow \exists\, ioB_1 : ioData[LocalB_1, Name, Value]; & [\text{def}^n \text{ of } \preceq] \\
\qquad \ldots;\ ioB_N : ioData[LocalB_N, Name, Value] \bullet & \\
\quad ioA \preceq_{fwd} ioB_1 \vee ioA \preceq_{bkwd} ioB_1 & \\
\quad \wedge & \\
\quad \forall\, i : 1 \mathinner{..} N-1 \bullet\ ioB_i \preceq_{fwd} ioB_{i+1} \vee ioB_i \preceq_{bkwd} ioB_{i+1} & \\
\quad \wedge & \\
\quad ioB_N \preceq_{fwd} ioC \vee ioB_N \preceq_{bkwd} ioC & \\
\Rightarrow \exists\, ioB_1 : ioData[LocalB_1, Name, Value]; & [\text{Lemmas 7 and 8}] \\
\qquad \ldots;\ ioB_N : ioData[LocalB_N, Name, Value] \bullet & \\
\quad ioA \sqsubseteq ioB_1 & \\
\quad \wedge & \\
\quad \forall\, i : 1 \mathinner{..} N-1 \bullet\ ioB_i \sqsubseteq ioB_{i+1} & \\
\quad \wedge & \\
\quad ioB_N \sqsubseteq ioC & \\
\Rightarrow ioA \sqsubseteq ioC & [\text{transitivity of } \sqsubseteq]
\end{array}$$

We have proved the soundness of the simulation rules for ioData types, that is, if one ioData type simulates a second, then the first is refined by the second.

3.7 Preservation of refinement

Our desired and expected theorem about ioData types – and conclusion to this document – is that the semantic refinement of ioData types is equivalent to the failures refinement of their equivalent ioProcesses.

Theorem 5 Given two ioData types, the first is refined by the second precisely when the ioProcess corresponding to the first is refined by the ioProcess corresponding to the second. □

This result is proved in Chapter C in the Appendix.

Appendix A

Relaxing the simulation rules for ioData types

In this section we present the proofs of the equivalence of the totalised and relaxed versions of the simulation rules for ioData types.

A.1 Forwards simulations rules

In this section we prove that the relaxed and totalised versions of the forwards simulation rules for ioData types are equivalent

Theorem 6 Let $ioA : ioData[LocalA, Name, Value]$ and $ioC : ioData[LocalC, Name, Value]$ be ioData types such that there exist relations r and R as defined below.

$$r : LocalA \leftrightarrow LocalC$$

and

$$\begin{array}{l}
R : (\, LocalA \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \leftrightarrow \\
\qquad (\, LocalC \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \\
\hline
R = \{a : LocalA;\ c : LocalC;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad a \mapsto c \in r \bullet (a, ins, outs) \mapsto (c, ins, outs)\}
\end{array}$$

Then Rules A.1, A.2 and A.3 below

$$startBatch^T\ ioC \subseteq startBatch^T\ ioA \,⨾\, R^T \tag{A.1}$$

$$R^T \fatsemi stopBatch^T\ ioC \ \subseteq\ stopBatch^T\ ioA \tag{A.2}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\ ioOps\ ioC \bullet \\ \quad R^T \fatsemi batch^T\ ioOps\ ioC\ n \ \subseteq\ batch^T\ ioOps\ ioA\ n \fatsemi R^T \end{array} \tag{A.3}$$

hold precisely when Rules A.4, A.5, A.6, A.7, A.8 and A.9 hold.

$$ioInit\ ioC \ \subseteq\ ioInit\ ioA \fatsemi (r \parallel \mathrm{id}) \tag{A.4}$$

$$(r \parallel \mathrm{id}) \fatsemi ioFinal\ ioC \ \subseteq\ ioFinal\ ioA \tag{A.5}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\ ioOps\ ioC \bullet \\ \quad (r \parallel \mathrm{id}) \fatsemi ioOps\ ioC\ n \ \subseteq\ ioOps\ ioA\ n \fatsemi (r \parallel \mathrm{id}) \end{array} \tag{A.6}$$

$$\mathrm{dom}\ ioInit\ ioA \ =\ \mathrm{dom}\ ioInit\ ioC \tag{A.7}$$

$$(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \ \subseteq\ \mathrm{dom}\ ioFinal\ ioC \tag{A.8}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\ ioOps\ ioC \bullet \\ \quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \ \subseteq\ \mathrm{dom}\ ioOps\ ioC\ n \end{array} \tag{A.9}$$

□

Proof

By Lemmas 1, 2 and 3.

Lemma 1

$$\begin{array}{l} startBatch^T\ ioC \subseteq startBatch^T\ ioA \fatsemi R^T \\ \quad \Leftrightarrow (\ ioInit\ ioC \subseteq ioInit\ ioA \fatsemi (r \parallel \mathrm{id}) \\ \qquad \wedge \\ \qquad \mathrm{dom}\ ioInit\ ioA = \mathrm{dom}\ ioInit\ ioC\ ) \end{array}$$

□

$$\begin{array}{ll} startBatch^T\ ioC \subseteq startBatch^T\ ioA \fatsemi R^T & \\ & \\ \quad \Leftrightarrow startBatch^T\ ioC \rhd BatchState[LocalC,\ Value] & [\text{property of } \subseteq] \\ \qquad\qquad \subseteq & \\ \qquad\qquad startBatch^T\ ioA \fatsemi R^T \rhd BatchState[LocalA,\ Value] & \\ \qquad \wedge & \\ \qquad startBatch^T\ ioC \rhd BatchState[\{\perp\},\ Value] & \\ \qquad\qquad \subseteq & \\ \qquad\qquad startBatch^T\ ioA \fatsemi R^T \rhd BatchState[\{\perp\},\ Value] & \end{array}$$

$$\begin{array}{ll}
\Leftrightarrow startBatch^{T}\ ioC \rhd BatchState[LocalC, Value] & [startBatch^{T}] \\
\qquad \subseteq & \\
\qquad startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[LocalA, Value] & \\
\quad \wedge & \\
\quad \{inI : Value;\ ins : \mathrm{seq}\ Value \mid inI \notin \mathrm{dom}\ ioInit\ ioC \bullet & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (\bot, ins, \langle\rangle)\} & \\
\qquad \subseteq & \\
\qquad startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[\{\bot\}, Value] &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow startBatch^{T}\ ioC \rhd BatchState[LocalC, Value] & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[LocalA, Value] & \\
\quad \wedge & \\
\quad \forall inI : Value;\ ins : \mathrm{seq}\ Value \bullet & \\
\qquad inI \notin \mathrm{dom}\ ioInit\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (\bot, ins, \langle\rangle) & \\
\qquad\qquad\quad \in startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[\{\bot\}, Value] &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow startBatch^{T}\ ioC \rhd BatchState[LocalC, Value] & [\mathrm{def}^{n} \text{ of } R^{T}] \\
\qquad \subseteq & \\
\qquad startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[LocalA, Value] & \\
\quad \wedge & \\
\quad \forall inI : Value;\ ins : \mathrm{seq}\ Value \bullet & \\
\qquad inI \notin \mathrm{dom}\ ioInit\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (\bot, ins, \langle\rangle) & \\
\qquad\qquad\quad \in startBatch^{T}\ ioA \fatsemi \mathrm{id}[BatchState[\{\bot\}, Value]] &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow startBatch^{T}\ ioC \rhd BatchState[LocalC, Value] & [\text{property of id}] \\
\qquad \subseteq & \\
\qquad startBatch^{T}\ ioA \fatsemi R^{T} \rhd BatchState[LocalA, Value] & \\
\quad \wedge & \\
\quad \forall inI : Value;\ ins : \mathrm{seq}\ Value \bullet & \\
\qquad inI \notin \mathrm{dom}\ ioInit\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (\bot, ins, \langle\rangle) & \\
\qquad\qquad\quad \in startBatch^{T}\ ioA \rhd BatchState[\{\bot\}, Value] &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \rhd \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathit{startBatch}^T] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \,⨾\, R^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \forall \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \rhd \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \,⨾\, R^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \rhd \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } R^T] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \,⨾\, R & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{startBatch}\ \mathit{ioC} & [\mathrm{def}^n \text{ of } \mathit{startBatch}^T] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \,⨾\, R & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \{\mathit{ins} : \mathrm{seq}\ \mathit{Value};\ c' : \mathit{LocalC};\ \mathit{inI}, \mathit{outI} : \mathit{Value} \mid & [\mathit{startBatch}] \\
\qquad\quad \mathit{inI} \mapsto (c', \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \bullet & \\
\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c', \mathit{ins}, \langle \mathit{outI} \rangle\}) & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \,⨾\, R & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall \mathit{ins} : \mathrm{seq}\ \mathit{Value};\ c' : \mathit{LocalC};\ \mathit{inI}, \mathit{outI} : \mathit{Value} \bullet & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad \mathit{inI} \mapsto (c', \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c', \mathit{ins}, \langle \mathit{outI} \rangle) \in \mathit{startBatch}\ \mathit{ioA} \,⨾\, R & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, ins : \mathrm{seq}\, Value;\ c' : LocalC;\ inI, outI : Value \bullet \qquad [\mathrm{def}^n \text{ of } \fatsemi] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a' : LocalA;\ is, os : \mathrm{seq}\, Value \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (a', is, os) \in startBatch\ ioA \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a', is, os) \mapsto (c', ins, \langle outI \rangle) \in R \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, ins : \mathrm{seq}\, Value;\ c' : LocalC;\ inI, outI : Value \bullet \qquad [\mathrm{def}^n \text{ of } R] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a' : LocalA;\ is, os : \mathrm{seq}\, Value \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (a', is, os) \in startBatch\ ioA \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad a' \mapsto c' \in r \ \wedge\ is = ins \ \wedge\ os = \langle outI \rangle \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, ins : \mathrm{seq}\, Value;\ c' : LocalC;\ inI, outI : Value \bullet \qquad [\text{one point rule}] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a' : LocalA \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (a', ins, \langle outI \rangle) \in startBatch\ ioA \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad a' \mapsto c' \in r \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, ins : \mathrm{seq}\, Value;\ c' : LocalC;\ inI, outI : Value \bullet \qquad [startBatch] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a' : LocalA \bullet \\
\qquad\qquad\qquad inI \mapsto (a', outI) \in ioInit\ ioA \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad a' \mapsto c' \in r \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, ins : \operatorname{seq} Value;\ c' : LocalC;\ inI, outI : Value \bullet & [\operatorname{def}^{n} \text{ of } \|] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \exists\, a' : LocalA \bullet & \\
\qquad\qquad\qquad inI \mapsto (a', outI) \in ioInit\ ioA & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (a', outI) \mapsto (c', outI) \in (r \parallel \mathrm{id}) & \\
\quad \wedge & \\
\quad \operatorname{dom} ioInit\ ioA \subseteq \operatorname{dom} ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, ins : \operatorname{seq} Value;\ c' : LocalC;\ inI, outI : Value \bullet & [\operatorname{def}^{n} \text{ of } \exists] \\
\qquad inI \mapsto (c', outI) \in ioInit\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad inI \mapsto (c', outI) \in ioInit\ ioA \mathbin{⨾} (r \parallel \mathrm{id}) & \\
\quad \wedge & \\
\quad \operatorname{dom} ioInit\ ioA \subseteq \operatorname{dom} ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow ioInit\ ioC \subseteq ioInit\ ioA \mathbin{⨾} (r \parallel \mathrm{id}) & [\operatorname{def}^{n} \text{ of } \subseteq] \\
\quad \wedge & \\
\quad \operatorname{dom} ioInit\ ioA \subseteq \operatorname{dom} ioInit\ ioC &
\end{array}
$$

We have shown that the following two rules hold:

1. $ioInit\ ioC \subseteq ioInit\ ioA \mathbin{⨾} (r \parallel \mathrm{id})$
2. $\operatorname{dom} ioInit\ ioA \subseteq \operatorname{dom} ioInit\ ioC$

But, considering the domains of the relations, (1) implies that

$$\operatorname{dom} ioInit\ ioC \subseteq \operatorname{dom} ioInit\ ioA \mathbin{⨾} (r \parallel \mathrm{id})$$

or equivalently that

$$\operatorname{dom} ioInit\ ioC \subseteq \operatorname{dom}(ioInit\ ioA \rhd \operatorname{dom}(r \parallel \mathrm{id}))$$

and hence that

$$\operatorname{dom} ioInit\ ioC \subseteq \operatorname{dom} ioInit\ ioA$$

Observing that

$$
\begin{array}{l}
\operatorname{dom} ioInit\ ioC \subseteq \operatorname{dom} ioInit\ ioA \\
\wedge \\
\operatorname{dom} ioInit\ ioA \subseteq \operatorname{dom} ioInit\ ioC
\end{array}
$$

the second rule may be simplified to $\operatorname{dom} ioInit\ ioA = \operatorname{dom} ioInit\ ioC$ and hence the two forwards simulation rules on the initialisation operation are:

1. $ioInit\ ioC \subseteq ioInit\ ioA \mathbin{⨾} (r \parallel \mathrm{id})$
2. $\mathrm{dom}\ ioInit\ ioA = \mathrm{dom}\ ioInit\ ioC$

Lemma 2

$$
\begin{aligned}
& R^T \mathbin{⨾} stopBatch^T\ ioC \subseteq stopBatch^T\ ioA \\
& \quad \Leftrightarrow (\ (r \parallel \mathrm{id}) \mathbin{⨾} ioFinal\ ioC \subseteq ioFinal\ ioA \\
& \qquad\quad \wedge \\
& \qquad\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC\ )
\end{aligned}
$$

□

$$
\begin{aligned}
& R^T \mathbin{⨾} stopBatch^T\ ioC \subseteq stopBatch^T\ ioA \\
& \Leftrightarrow batchState[LocalA, Value] \lhd R^T \mathbin{⨾} stopBatch^T\ ioC && [\text{property of } \subseteq] \\
& \qquad \subseteq \\
& \qquad batchState[LocalA, Value] \lhd stopBatch^T\ ioA \\
& \quad \wedge \\
& \quad batchState[\{\bot\}, Value] \lhd R^T \mathbin{⨾} stopBatch^T\ ioC \\
& \qquad \subseteq \\
& \qquad batchState[\{\bot\}, Value] \lhd stopBatch^T\ ioA \\
& \Leftrightarrow batchState[LocalA, Value] \lhd R^T \mathbin{⨾} stopBatch^T\ ioC && [\mathrm{def}^n \text{ of } R^T] \\
& \qquad \subseteq \\
& \qquad batchState[LocalA, Value] \lhd stopBatch^T\ ioA \\
& \quad \wedge \\
& \quad \mathrm{id}[batchState[\{\bot\}, Value]] \mathbin{⨾} stopBatch^T\ ioC \\
& \qquad \subseteq \\
& \qquad batchState[\{\bot\}, Value] \lhd stopBatch^T\ ioA \\
& \Leftrightarrow batchState[LocalA, Value] \lhd R^T \mathbin{⨾} stopBatch^T\ ioC && [\text{property of id}] \\
& \qquad \subseteq \\
& \qquad batchState[LocalA, Value] \lhd stopBatch^T\ ioA \\
& \quad \wedge \\
& \quad batchState[\{\bot\}, Value] \lhd stopBatch^T\ ioC \\
& \qquad \subseteq \\
& \qquad batchState[\{\bot\}, Value] \lhd stopBatch^T\ ioA \\
& \Leftrightarrow batchState[LocalA, Value] \lhd R^T \mathbin{⨾} stopBatch^T\ ioC && [stopBatch^T] \\
& \qquad \subseteq \\
& \qquad batchState[LocalA, Value] \lhd stopBatch^T\ ioA \\
& \quad \wedge \\
& \quad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\bot, \langle in \rangle, outs) \mapsto outs\} \\
& \qquad \subseteq \\
& \qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\bot, \langle in \rangle, outs) \mapsto outs\}
\end{aligned}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd R^T \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC} & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} & \\
\quad \wedge & \\
\quad \text{true} & \\
& \\
\Leftrightarrow \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd R^T \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC} & [\text{property of } \wedge] \\
\qquad \subseteq & \\
\qquad \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} & \\
& \\
\Leftrightarrow R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC} & [\text{def}^n \text{ of } R^T] \\
\qquad \subseteq & \\
\qquad \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} & \\
& \\
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\quad \wedge & \\
\quad (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \cap \mathit{third} & \\
\qquad \subseteq & \\
\qquad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \cap \mathit{third} & \\
& \\
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & [\text{def}^n \text{ of } \subseteq] \\
\qquad \subseteq & \\
\qquad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\quad \wedge & \\
\quad \forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \text{seq}\ \mathit{Value} \bullet & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \cap \mathit{third} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' & \\
\qquad\qquad \in & \\
\qquad\qquad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) & \\
\qquad\qquad \cap & \\
\qquad\qquad \mathit{third} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & \text{[property of } \cap]\\
\quad \subseteq & \\
\quad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\wedge & \\
\forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (\, (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) & \\
\qquad \wedge & \\
\quad\ (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{third}\,) & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\, (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' & \\
\qquad\qquad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} & \\
\qquad\qquad \wedge & \\
\qquad\quad\ (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{third}\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & \text{[property of } \Rightarrow]\\
\quad \subseteq & \\
\quad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\wedge & \\
\forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{third}\,) \bullet & \\
\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) & \\
\qquad \Rightarrow & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' & \\
\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & [\text{def}^n \text{ of } \mathit{third}]\\
\quad \subseteq & \\
\quad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\wedge & \\
\forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \mathrm{seq}\ \mathit{Value} \mid \mathit{os} = \mathit{os}' \bullet & \\
\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) & \\
\qquad \Rightarrow & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' & \\
\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) \setminus \mathit{third} & \text{[one point rule]}\\
\quad \subseteq & \\
\quad (\, \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) \setminus \mathit{third} & \\
\wedge & \\
\forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in (\, R \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}\,) & \\
\qquad \Rightarrow & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os} & \\
\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA} &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \mathbin{;} stopBatch^T\ ioC\,) \setminus third \qquad [\text{def}^n \text{ of } \exists] \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ is, os, : \text{seq } Value \bullet \\
\qquad (\, \exists\, c : LocalC;\ ins, outs : \text{seq } Value \bullet \\
\qquad\qquad\quad (a, is, os) \mapsto (c, ins, outs) \in R \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, ins, outs) \mapsto os \\
\qquad\qquad\qquad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioC\,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, is, os) \mapsto os \\
\qquad\qquad\quad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \mathbin{;} stopBatch^T\ ioC\,) \setminus third \qquad [\text{def}^n \text{ of } R] \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ is, os, : \text{seq } Value \bullet \\
\qquad (\, \exists\, c : LocalC;\ ins, outs : \text{seq } Value \bullet \\
\qquad\qquad\quad a \mapsto c \in r \ \wedge\ ins = is \ \wedge\ outs = os \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, ins, outs) \mapsto os \\
\qquad\qquad\qquad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioC\,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, is, os) \mapsto os \\
\qquad\qquad\quad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \mathbin{;} stopBatch^T\ ioC\,) \setminus third \qquad [\text{one point rule}] \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ is, os, : \text{seq } Value \bullet \\
\qquad (\, \exists\, c : LocalC \bullet \\
\qquad\qquad\quad a \mapsto c \in r \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, is, os) \mapsto os \\
\qquad\qquad\qquad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioC\,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, is, os) \mapsto os \\
\qquad\qquad\quad \in batchState[LocalA, Value] \lhd stopBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\ R \fatsemi stopBatch^{T}\ ioC\ ) \setminus third \qquad\qquad [\mathrm{def}^{n} \text{ of } stopBatch^{T}] \\
\qquad \subseteq \\
\qquad (\ batchState[LocalA, Value] \lhd stopBatch^{T}\ ioA\ ) \setminus third \\
\quad \land \\
\quad \forall\ a : LocalA;\ is, os, : \mathrm{seq}\ Value \bullet \\
\qquad (\ \exists\ c : LocalC \bullet \\
\qquad\qquad\qquad a \mapsto c \in r \\
\qquad\qquad\qquad \land \\
\qquad\qquad\qquad (c, is, os) \mapsto os \\
\qquad\qquad\qquad\quad \in (\ stopBatch\ ioC \\
\qquad\qquad\qquad\qquad \cup \\
\qquad\qquad\qquad\qquad \{\ local : LocalC;\ in : Value;\ outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioC \bullet \\
\qquad\qquad\qquad\qquad\qquad\quad (local, \langle in \rangle, outs) \mapsto outs\ \}\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, is, os) \mapsto os \\
\qquad\qquad\quad \in (\ stopBatch\ ioA \\
\qquad\qquad\qquad \cup \\
\qquad\qquad\qquad \{\ local : LocalA;\ in : Value;\ outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad\qquad\quad (local, \langle in \rangle, outs) \mapsto outs\ \}\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\ R \fatsemi stopBatch^{T}\ ioC\ ) \setminus third \qquad\qquad [\mathrm{def}^{n} \text{ of } stopBatch] \\
\qquad \subseteq \\
\qquad (\ batchState[LocalA, Value] \lhd stopBatch^{T}\ ioA\ ) \setminus third \\
\quad \land \\
\quad \forall\ a : LocalA;\ is, os, : \mathrm{seq}\ Value \bullet \\
\qquad (\ \exists\ c : LocalC \bullet \\
\qquad\qquad\qquad a \mapsto c \in r \\
\qquad\qquad\qquad \land \\
\qquad\qquad\qquad (c, is, os) \mapsto os \\
\qquad\qquad\qquad\quad \in \{\ local : LocalC;\ in : Value;\ outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioC \bullet \\
\qquad\qquad\qquad\qquad\qquad\quad (local, \langle in \rangle, outs) \mapsto outs\ \}\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, is, os) \mapsto os \\
\qquad\qquad\quad \in \{\ local : LocalA;\ in : Value;\ outs : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad\qquad\quad (local, \langle in \rangle, outs) \mapsto outs\ \}\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \comp stopBatch^T\ ioC \,) \setminus third \qquad \text{[set theory]} \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA \,) \setminus third \\
\quad \land \\
\quad \forall\, a : LocalA;\ is, os, : \mathrm{seq}\ Value \bullet \\
\qquad (\, \exists\, c : LocalC \bullet \\
\qquad\qquad\qquad a \mapsto c \in r \\
\qquad\qquad\qquad \land \\
\qquad\qquad\qquad \#is = 1 \ \land\ (c, head\ is) \notin \mathrm{dom}\ ioFinal\ ioC \,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \#is = 1 \ \land\ (a, head\ is) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \comp stopBatch^T\ ioC \,) \setminus third \qquad \text{[property of } \exists\text{]} \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA \,) \setminus third \\
\quad \land \\
\quad \forall\, a : LocalA;\ is, os, : \mathrm{seq}\ Value \bullet \\
\qquad (\, \#is = 1 \\
\qquad\ \ \land \\
\qquad\ \ \exists\, c : LocalC \bullet a \mapsto c \in r \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (c, head\ is) \notin \mathrm{dom}\ ioFinal\ ioC \,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \#is = 1 \ \land\ (a, head\ is) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\, R \comp stopBatch^T\ ioC \,) \setminus third \qquad \text{[property of } \forall\text{]} \\
\qquad \subseteq \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA \,) \setminus third \\
\quad \land \\
\quad \forall\, a : LocalA;\ is : \mathrm{seq}\ Value \mid \#is = 1 \bullet \\
\qquad (\, \exists\, c : LocalC \bullet a \mapsto c \in r \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (c, head\ is) \notin \mathrm{dom}\ ioFinal\ ioC \,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, head\ is) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \operatorname{;} stopBatch^T\ ioC\,) \setminus third & \text{[property of } \Rightarrow\text{]} \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \wedge & \\
\quad \forall\, a : LocalA;\ is : \operatorname{seq} Value \mid \#is = 1 \bullet & \\
\qquad (a, head\ is) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \neg\, (\exists\, c : LocalC \bullet a \mapsto c \in r \wedge (c, head\ is) \notin \operatorname{dom} ioFinal\ ioC\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \operatorname{;} stopBatch^T\ ioC\,) \setminus third & \text{[property of } \forall\text{]} \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \wedge & \\
\quad \forall\, a : LocalA;\ is : \operatorname{seq} Value \mid \#is = 1 \bullet & \\
\qquad (a, head\ is) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet \neg\, (\, a \mapsto c \in r \wedge (c, head\ is) \notin \operatorname{dom} ioFinal\ ioC\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \operatorname{;} stopBatch^T\ ioC\,) \setminus third & \text{[de Morgan's]} \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \wedge & \\
\quad \forall\, a : LocalA;\ is : \operatorname{seq} Value \mid \#is = 1 \bullet & \\
\qquad (a, head\ is) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet a \mapsto c \notin r \vee (c, head\ is) \in \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \operatorname{;} stopBatch^T\ ioC\,) \setminus third & \text{[def}^n \text{ of } \mathit{implies}\text{]} \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \wedge & \\
\quad \forall\, a : LocalA;\ is : \operatorname{seq} Value \mid \#is = 1 \bullet & \\
\qquad (a, head\ is) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet a \mapsto c \in r \Rightarrow (c, head\ is) \in \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \fatsemi stopBatch^T\ ioC\,) \setminus third & [\text{property of } \#] \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \land & \\
\quad \forall\, a : LocalA;\ in : Value \bullet & \\
\qquad (a, in) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet a \mapsto c \in r \Rightarrow (c, in) \in \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \fatsemi stopBatch^T\ ioC\,) \setminus third & [\text{property of id}] \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \land & \\
\quad \forall\, a : LocalA;\ in : Value \bullet & \\
\qquad (a, in) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet (a, in) \mapsto (c, in) \in (r \parallel \mathrm{id}) & \\
\qquad\qquad\qquad\qquad \Rightarrow & \\
\qquad\qquad\qquad\qquad (c, in) \in \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \fatsemi stopBatch^T\ ioC\,) \setminus third & [\text{def}^n \text{ of } (\!|\ |\!)] \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \land & \\
\quad \forall\, a : LocalA;\ in : Value \bullet & \\
\qquad (a, in) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \forall\, c : LocalC \bullet (c, in) \in (r \parallel \mathrm{id})(\!|\ \{(a, in)\}\ |\!) & \\
\qquad\qquad\qquad\qquad \Rightarrow & \\
\qquad\qquad\qquad\qquad (c, in) \in \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \fatsemi stopBatch^T\ ioC\,) \setminus third & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \land & \\
\quad \forall\, a : LocalA;\ in : Value \bullet & \\
\qquad (a, in) \in \operatorname{dom} ioFinal\ ioA & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (r \parallel \mathrm{id})(\!|\ \{(a, in)\}\ |\!) \subseteq \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\, R \fatsemi stopBatch^T\ ioC\,) \setminus third & [\text{property of } \subseteq] \\
\qquad \subseteq & \\
\qquad (\, batchState[LocalA, Value] \lhd stopBatch^T\ ioA\,) \setminus third & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} ioFinal\ ioA\ |\!) \subseteq \operatorname{dom} ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \bullet \qquad [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad (a, is, os) \mapsto os' \in (\ R \circ stopBatch^T\ ioC\ ) \setminus third \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, is, os) \mapsto os' \\
\qquad\qquad \in \\
\qquad\qquad (\ batchState[LocalA, Value] \lhd stopBatch^T\ ioA\ ) \setminus third \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \bullet \qquad [\text{property of } \setminus] \\
\qquad (\ (a, is, os) \mapsto os' \in (\ R \circ stopBatch^T\ ioC\ ) \\
\qquad\ \wedge \\
\qquad\ (a, is, os) \mapsto os' \notin third\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (\ (a, is, os) \mapsto os' \\
\qquad\qquad \in (\ batchState[LocalA, Value] \lhd stopBatch^T\ ioA\ ) \\
\qquad\quad\ \wedge \\
\qquad\quad\ (a, is, os) \mapsto os' \notin third\ ) \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value\ | \qquad [\text{property of } \Rightarrow] \\
\qquad\qquad\qquad (a, is, os) \mapsto os' \notin third \bullet \\
\qquad (a, is, os) \mapsto os' \in (\ R \circ stopBatch^T\ ioC\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, is, os) \mapsto os' \\
\qquad\qquad \in (\ batchState[LocalA, Value] \lhd stopBatch^T\ ioA\ ) \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value\ |\ os' \neq os \bullet \qquad [\mathrm{def}^n \text{ of } third] \\
\qquad (a, is, os) \mapsto os' \in (\ R \circ stopBatch^T\ ioC\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, is, os) \mapsto os' \\
\qquad\qquad \in (\ batchState[LocalA, Value] \lhd stopBatch^T\ ioA\ ) \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \operatorname{seq} \mathit{Value} \mid \mathit{os}' \neq \mathit{os} \bullet \qquad [\text{def}^{\,n} \text{ of } \exists] \\
\qquad (\exists\, c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs} : \operatorname{seq} \mathit{Value} \bullet \\
\qquad\qquad (a, \mathit{is}, \mathit{os}) \mapsto (c, \mathit{ins}, \mathit{outs}) \in R \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{os}' \\
\qquad\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC}\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \\
\qquad\qquad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioA} \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}\ |\!) \subseteq \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \operatorname{seq} \mathit{Value} \mid \mathit{os}' \neq \mathit{os} \bullet \qquad [\text{def}^{\,n} \text{ of } R] \\
\qquad (\exists\, c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs} : \operatorname{seq} \mathit{Value} \bullet \\
\qquad\qquad a \mapsto c \in r\ \wedge\ \mathit{ins} = \mathit{is}\ \wedge\ \mathit{outs} = \mathit{os} \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{os}' \\
\qquad\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC}\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \\
\qquad\qquad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioA} \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}\ |\!) \subseteq \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os}, \mathit{os}' : \operatorname{seq} \mathit{Value} \mid \mathit{os}' \neq \mathit{os} \bullet \qquad [\text{one point rule}] \\
\qquad (\exists\, c : \mathit{LocalC} \bullet \\
\qquad\qquad a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \\
\qquad\qquad\quad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC}\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \\
\qquad\qquad \in \mathit{batchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioA} \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}\ |\!) \subseteq \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \mid os' \neq os \bullet & [\mathrm{def}^{\,n} \text{ of } stopBatch^{\,T}] \\
\quad (\exists\, c : LocalC \bullet & \\
\qquad\quad a \mapsto c \in r & \\
\qquad\quad \wedge & \\
\qquad\quad (c, is, os) \mapsto os' & \\
\qquad\qquad \in (\ stopBatch\ ioC & \\
\qquad\qquad\quad \cup & \\
\qquad\qquad\quad \{local : LocalC;\ in : Value;\ outs : \mathrm{seq}\ Value \mid & \\
\qquad\qquad\qquad (local, in) \notin \mathrm{dom}\ ioFinal\ ioC \bullet & \\
\qquad\qquad\qquad\quad (local, \langle in \rangle, outs) \mapsto outs\}\ ) & \\
\qquad \Rightarrow & \\
\qquad (a, is, os) \mapsto os' & \\
\qquad\quad \in (\ stopBatch\ ioA & \\
\qquad\qquad \cup & \\
\qquad\qquad \{local : LocalA;\ in : Value;\ outs : \mathrm{seq}\ Value \mid & \\
\qquad\qquad\quad (local, in) \notin \mathrm{dom}\ ioFinal\ ioA \bullet & \\
\qquad\qquad\qquad (local, \langle in \rangle, outs) \mapsto outs\}\ ) & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \mid os' \neq os \bullet & [os' \neq os] \\
\quad (\exists\, c : LocalC \bullet & \\
\qquad\quad a \mapsto c \in r & \\
\qquad\quad \wedge & \\
\qquad\quad (c, is, os) \mapsto os' \in stopBatch\ ioC & \\
\qquad \Rightarrow & \\
\qquad (a, is, os) \mapsto os' \in stopBatch\ ioA & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \mid os' \neq os \bullet & [\mathrm{def}^{\,n} \text{ of } stopBatch] \\
\quad (\exists\, c : LocalC \bullet & \\
\qquad\quad a \mapsto c \in r & \\
\qquad\quad \wedge & \\
\qquad\quad \#is = 1\ \wedge\ os = first\ os' & \\
\qquad\quad \wedge & \\
\qquad\quad (c, head\ is) \mapsto last\ os' \in ioFinal\ ioC & \\
\qquad \Rightarrow & \\
\qquad \#is = 1\ \wedge\ os = first\ os' & \\
\qquad \wedge & \\
\qquad (a, head\ is) \mapsto last\ os' \in ioFinal\ ioA & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ is, os, os' : \mathrm{seq}\ Value \mid & [\text{property of } \mid] \\
\qquad \#is = 1\ \land\ os = \mathit{first}\ os' \bullet & \\
\quad (\exists\, c : LocalC \bullet a \mapsto c \in r & \\
\qquad \land & \\
\qquad (c, head\ is) \mapsto last\ os' \in ioFinal\ ioC & \\
\quad \Rightarrow & \\
\quad (a, head\ is) \mapsto last\ os' \in ioFinal\ ioA & \\
\land & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF : Value;\ os, os' : \mathrm{seq}\ Value \mid & [\text{property of } \#] \\
\qquad os = \mathit{first}\ os' \bullet & \\
\quad (\exists\, c : LocalC \bullet a \mapsto c \in r & \\
\qquad \land & \\
\qquad (c, inF) \mapsto last\ os' \in ioFinal\ ioC & \\
\quad \Rightarrow & \\
\quad (a, inF) \mapsto last\ os' \in ioFinal\ ioA & \\
\land & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF, outF : Value;\ os, os' : \mathrm{seq}\ Value \mid & [\mathrm{def}^n \text{ of } \mathit{first}] \\
\qquad os' = os \frown \langle outF \rangle \bullet & \\
\quad (\exists\, c : LocalC \bullet a \mapsto c \in r & \\
\qquad \land & \\
\qquad (c, inF) \mapsto last\ os' \in ioFinal\ ioC & \\
\quad \Rightarrow & \\
\quad (a, inF) \mapsto last\ os' \in ioFinal\ ioA & \\
\land & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF, outF : Value;\ os : \mathrm{seq}\ Value \mid & [\text{one point rule}] \\
\qquad \bullet & \\
\quad (\exists\, c : LocalC \bullet a \mapsto c \in r & \\
\qquad \land & \\
\qquad (c, inF) \mapsto last\,(os \frown \langle outF \rangle) \in ioFinal\ ioC & \\
\quad \Rightarrow & \\
\quad (a, inF) \mapsto last\,(os \frown \langle outF \rangle) \in ioFinal\ ioA & \\
\land & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioFinal\ ioA\ |\!) \subseteq \mathrm{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF, outF : Value \bullet & [\text{def}^{\,n} \text{ of } last] \\
\qquad (\exists\, c : LocalC \bullet a \mapsto c \in r & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, inF) \mapsto outF \in ioFinal\ ioA & \\
\quad \wedge & \\
\quad (r \parallel \text{id})(\!|\ \text{dom}\ ioFinal\ ioA\ |\!) \subseteq \text{dom}\ ioFinal\ ioC &
\end{array}$$

$\Leftrightarrow$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF, outF : Value \bullet & [\text{def}^{\,n} \text{ of id}] \\
\qquad (\exists\, c : LocalC \bullet (a, inF) \mapsto (c, inF) \in (r \parallel \text{id}) & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, inF) \mapsto outF \in ioFinal\ ioA & \\
\quad \wedge & \\
\quad (r \parallel \text{id})(\!|\ \text{dom}\ ioFinal\ ioA\ |\!) \subseteq \text{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ inF, outF : Value \bullet & [\text{property of } \fatsemi] \\
\qquad (a, inF) \mapsto outF \in (r \parallel \text{id}) \fatsemi ioFinal\ ioC & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, inF) \mapsto outF \in ioFinal\ ioA & \\
\quad \wedge & \\
\quad (r \parallel \text{id})(\!|\ \text{dom}\ ioFinal\ ioA\ |\!) \subseteq \text{dom}\ ioFinal\ ioC &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow (r \parallel \text{id}) \fatsemi ioFinal\ ioC \subseteq ioFinal\ ioA & [\text{def}^{\,n} \text{ of } \subseteq] \\
\quad \wedge & \\
\quad (r \parallel \text{id})(\!|\ \text{dom}\ ioFinal\ ioA\ |\!) \subseteq \text{dom}\ ioFinal\ ioC &
\end{array}$$

Lemma 3

$$\begin{array}{l}
\forall\, n : \text{dom}\ ioOps\ ioC \bullet \\
\qquad\qquad R^T \fatsemi batch^T\ ioOps\ ioC\ n\ \subseteq\ batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\quad \Leftrightarrow (\,(r \parallel \text{id}) \fatsemi ioOps\ ioC\ n \subseteq ioOps\ ioA\ n \fatsemi (r \parallel \text{id}) \\
\qquad \wedge \\
\qquad (r \parallel \text{id})(\!|\ \text{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \text{dom}\ ioOps\ ioC\ n
\end{array}$$

$\Box$

$$\begin{array}{l}
R^T \fatsemi batch^T\ ioOps\ ioC\ n \\
\quad \subseteq \\
\quad batch^T\ ioOps\ ioA\ n \fatsemi R^T
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{property of } \subseteq] \\
\quad \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\land & \\
\mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{def}^n \text{ of } R^T] \\
\quad \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\land & \\
\mathrm{id}[\mathit{BatchState}[\{\bot\}, \mathit{Value}]] \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{def}^n \text{ of id}] \\
\quad \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\land & \\
\mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{def}^n \text{ of } \mathit{batch}^T] \\
\quad \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\land & \\
\{ \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (\bot, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto (\bot, \mathit{ins}, \mathit{outs})\} & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{property of } \subseteq] \\
\quad \lhd R^T \mathbin{\S} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T & \\
\land & \\
\forall \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (\bot, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto (\bot, \mathit{ins}, \mathit{outs})\} & \\
\qquad \in & \\
\qquad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{\S} R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \fatsemi] \\
\qquad \lhd R^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \fatsemi R^T & \\
\wedge & \\
\forall\, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad \exists\, a : \mathit{Local}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\,(\perp, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto (a, \mathit{is}, \mathit{os}) & \\
\qquad\quad \in \mathit{BatchState}[\{\perp\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\qquad \wedge & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto (\perp, \mathit{ins}, \mathit{outs}) \in R^T\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \mathit{batch}^T] \\
\qquad \lhd R^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \fatsemi R^T & \\
\wedge & \\
\forall\, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad \exists\, a : \mathit{Local}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\,(\perp, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto (a, \mathit{is}, \mathit{os}) & \\
\qquad\quad \in \{\mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad\qquad (\perp, \langle \mathit{in} \rangle \frown \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}, \mathit{outs})\} & \\
\qquad\quad \wedge & \\
\qquad\quad (a, \mathit{is}, \mathit{os}) \mapsto (\perp, \mathit{ins}, \mathit{outs}) \in R^T\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{set theory}] \\
\qquad \lhd R^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \fatsemi R^T & \\
\wedge & \\
\forall\, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad \exists\, a : \mathit{Local}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\ a = \perp\ \wedge\ \mathit{is} = \mathit{ins}\ \wedge\ \mathit{os} = \mathit{outs} & \\
\qquad \wedge & \\
\qquad (a, \mathit{is}, \mathit{os}) \mapsto (\perp, \mathit{ins}, \mathit{outs}) \in R^T\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{one point rule}] \\
\qquad \lhd R^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\quad \subseteq & \\
\quad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \fatsemi R^T & \\
\wedge & \\
\forall\, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}, \mathit{outs}) \in R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow BatchState[LocalA, Value] & [\mathrm{def}^n \text{ of } R^T] \\
\qquad \lhd R^T \mathbin{;} batch^T\ ioOps\ ioC\ n & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T & \\
\wedge & \\
\text{true} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow BatchState[LocalA, Value] & [\text{property of } \wedge] \\
\qquad \lhd R^T \mathbin{;} batch^T\ ioOps\ ioC\ n & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } R^T] \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n & [\text{property of } \subseteq] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T & \\
\qquad\quad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
R \mathbin{;} batch^T\ ioOps\ ioC\ n & \\
\qquad \rhd BatchState[\{\bot\}, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T & \\
\qquad\quad \rhd BatchState[\{\bot\}, Value] &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T & \\
\qquad\quad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet & \\
\quad (a, ins, outs) \mapsto (\bot, ins', outs') \in R \mathbin{;} batch^T\ ioOps\ ioC\ n & \\
\qquad \Rightarrow & \\
\qquad (a, ins, outs) \mapsto (\bot, ins', outs') \in batch^T\ ioOps\ ioA\ n \mathbin{;} R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n & [\text{def}^n \text{ of } \fatsemi] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T & \\
\qquad\quad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ ins, outs, ins', outs' : \text{seq}\ Value \bullet & \\
\quad (\exists c : LocalC;\ is, os : \text{seq}\ Value \bullet & \\
\qquad\quad (a, ins, outs) \mapsto (c, is, os) \in R & \\
\qquad\quad \wedge & \\
\qquad\quad (c, is, os) \mapsto (\perp, ins', outs') \in batch^T\ ioOps\ ioC\ n\ ) & \\
\qquad \Rightarrow & \\
\qquad (a, ins, outs) \mapsto (\perp, ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n & [\text{def}^n \text{ of } R] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T & \\
\qquad\quad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ ins, outs, ins', outs' : \text{seq}\ Value \bullet & \\
\quad (\exists c : LocalC;\ is, os : \text{seq}\ Value \bullet & \\
\qquad\quad a \mapsto c \in r\ \wedge\ is = ins\ \wedge\ os = outs & \\
\qquad\quad \wedge & \\
\qquad\quad (c, is, os) \mapsto (\perp, ins', outs') \in batch^T\ ioOps\ ioC\ n\ ) & \\
\qquad \Rightarrow & \\
\qquad (a, ins, outs) \mapsto (\perp, ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R^T &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n & [\text{one point rule}] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T & \\
\qquad\quad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ ins, outs, ins', outs' : \text{seq}\ Value \bullet & \\
\quad (\exists c : LocalC \bullet a \mapsto c \in r & \\
\qquad\qquad \wedge & \\
\qquad\qquad (c, ins, outs) \mapsto (\perp, ins', outs') & \\
\qquad\qquad\quad \in & \\
\qquad\qquad\quad batch^T\ ioOps\ ioC\ n\ ) & \\
\qquad \Rightarrow & \\
\qquad (a, ins, outs) \mapsto (\perp, ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R^T &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \hfill [\mathrm{def}^n \text{ of } batch^T] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet \\
\qquad (\exists\, c : LocalC \bullet a \mapsto c \in r\ \wedge (c, head\ ins) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad ins' = tail\ ins \wedge outs' = outs\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (a, ins, outs) \mapsto (\bot, ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R^T
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \hfill [\text{property of } \Rightarrow] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad ins' = tail\ ins \wedge outs' = outs \bullet \\
\qquad\quad \exists\, c : LocalC \bullet a \mapsto c \in r \wedge (c, head\ ins) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, ins, outs) \mapsto (\bot, ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R^T
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \hfill [\mathrm{def}^n \text{ of } tail] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ in : Value;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad ins = \langle in \rangle \frown ins' \wedge outs' = outs \bullet \\
\qquad\quad \exists\, c : LocalC \bullet a \mapsto c \in r \wedge (c, head\ ins) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, ins, outs) \mapsto (\bot, ins', outs') \\
\qquad\qquad\qquad \in batch^T\ ioOps\ ioA\ n \fatsemi R^T
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\text{one point rule}]\\
\qquad\qquad \rhd BatchState[LocalC, Value]\\
\qquad \subseteq\\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T\\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value]\\
\quad \wedge\\
\quad \forall a : LocalA;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad \exists c : LocalC \bullet a \mapsto c \in r \wedge (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n\\
\qquad\qquad \Rightarrow\\
\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (\perp, ins', outs)\\
\qquad\qquad\qquad \in batch^T\ ioOps\ ioA\ n \fatsemi R^T
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\mathrm{def}^n \text{ of } \fatsemi]\\
\qquad\qquad \rhd BatchState[LocalC, Value]\\
\qquad \subseteq\\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T\\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value]\\
\quad \wedge\\
\quad \forall a : LocalA;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad \exists c : LocalC \bullet a \mapsto c \in r \wedge (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n\\
\qquad\qquad \Rightarrow\\
\qquad\qquad (\ \exists a' : LocalA_\perp;\ is, os : \mathrm{seq}\ Value \bullet\\
\qquad\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', is, os)\\
\qquad\qquad\qquad\qquad \in batch^T\ ioOps\ ioA\ n\\
\qquad\qquad\qquad \wedge\\
\qquad\qquad\qquad (a', is, os) \mapsto (\perp, ins', outs) \in R^T\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\mathrm{def}^n \text{ of } R^T]\\
\qquad\qquad \rhd BatchState[LocalC, Value]\\
\qquad \subseteq\\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T\\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value]\\
\quad \wedge\\
\quad \forall a : LocalA;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad \exists c : LocalC \bullet a \mapsto c \in r \wedge (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n\\
\qquad\qquad \Rightarrow\\
\qquad\qquad (\ \exists a' : LocalA_\perp;\ is, os : \mathrm{seq}\ Value \bullet\\
\qquad\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', is, os)\\
\qquad\qquad\qquad\qquad \in batch^T\ ioOps\ ioA\ n\\
\qquad\qquad\qquad \wedge\\
\qquad\qquad\qquad a' = \perp\ \wedge\ is = ins'\ \wedge\ os = outs\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n \hspace{12em} [\text{one point rule}] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet \\
\qquad \exists\, c : LocalC \bullet a \mapsto c \in r \wedge (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (\bot, ins', outs) \in batch^T\ ioOps\ ioA\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n \hspace{12em} [\mathrm{def}^n \text{ of } batch^T] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ in : Value \bullet \\
\qquad \exists\, c : LocalC \bullet a \mapsto c \in r \wedge (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, in) \notin \mathrm{dom}\ ioOps\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n \hspace{12em} [\text{property of } \Rightarrow] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall\, a : LocalA;\ in : Value \bullet \\
\qquad (a, in) \in \mathrm{dom}\ ioOps\ ioA\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \neg\, (\exists\, c : LocalC \bullet a \mapsto c \in r \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\text{property of } \forall] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall a : LocalA;\ in : Value \bullet \\
\qquad (a, in) \in \mathrm{dom}\ ioOps\ ioA\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \forall c : LocalC \bullet \neg\ (\ a \mapsto c \in r \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\text{de Morgan's}] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall a : LocalA;\ in : Value \bullet \\
\qquad (a, in) \in \mathrm{dom}\ ioOps\ ioA\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \forall c : LocalC \bullet\ a \mapsto c \notin r \\
\qquad\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad\qquad (c, in) \in \mathrm{dom}\ ioOps\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow R \fatsemi batch^T\ ioOps\ ioC\ n \qquad\qquad [\mathrm{def}^n \text{ of } \Rightarrow] \\
\qquad\qquad \rhd BatchState[LocalC, Value] \\
\qquad \subseteq \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \fatsemi R^T \\
\qquad\qquad\qquad \rhd BatchState[LocalC, Value] \\
\quad \wedge \\
\quad \forall a : LocalA;\ in : Value \bullet \\
\qquad (a, in) \in \mathrm{dom}\ ioOps\ ioA\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \forall c : LocalC \bullet\ a \mapsto c \in r \Rightarrow (c, in) \in \mathrm{dom}\ ioOps\ ioC
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{⨾} batch^T\ ioOps\ ioC\ n & \text{[property of id]} \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{⨾} R^T & \\
\qquad\qquad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ in : Value \bullet & \\
\quad (a, in) \in \operatorname{dom} ioOps\ ioA\ n & \\
\qquad \Rightarrow & \\
\qquad \forall c : LocalC \bullet (a, in) \mapsto (c, in) \in (r \parallel \mathrm{id}) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, in) \in \operatorname{dom} ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{⨾} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } (\!|\,|\!)] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{⨾} R^T & \\
\qquad\qquad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ in : Value \bullet & \\
\quad (a, in) \in \operatorname{dom} ioOps\ ioA\ n & \\
\qquad \Rightarrow & \\
\qquad \forall c : LocalC \bullet (c, in) \in (r \parallel \mathrm{id})(\!|\ \{(a, in)\}\ |\!) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, in) \in \operatorname{dom} ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{⨾} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{⨾} R^T & \\
\qquad\qquad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
\forall a : LocalA;\ in : Value \bullet & \\
\quad (a, in) \in \operatorname{dom} ioOps\ ioA\ n & \\
\qquad \Rightarrow & \\
\qquad (r \parallel \mathrm{id})(\!|\ \{(a, in)\}\ |\!) \subseteq \operatorname{dom} ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{⨾} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad \rhd BatchState[LocalC, Value] & \\
\quad \subseteq & \\
\quad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{⨾} R^T & \\
\qquad\qquad \rhd BatchState[LocalC, Value] & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \operatorname{dom} ioOps\ ioA\ n\ |\!) \subseteq \operatorname{dom} ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow R \mathbin{;} batch^T\ ioOps\ ioC\ n & [\mathrm{def}^n \text{ of } R^T] \\
\qquad\qquad \rhd BatchState[LocalC, Value] & \\
\qquad \subseteq & \\
\qquad BatchState[LocalA, Value] \lhd batch^T\ ioOps\ ioA\ n \mathbin{;} R & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad (a, ins, outs) \mapsto (c', ins', outs') \in R \mathbin{;} batch^T\ ioOps\ ioC\ n & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioA\ n \mathbin{;} R & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad (\exists\, c : LocalC;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad (a, ins, outs) \mapsto (c, is, os) \in R & \\
\qquad\qquad \land & \\
\qquad\qquad (c, is, os) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioC\ n\,) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioA\ n \mathbin{;} R & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } R] \\
\qquad (\exists\, c : LocalC;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad a \mapsto c \in r\ \land\ is = ins\ \land os = outs & \\
\qquad\qquad \land & \\
\qquad\qquad (c, is, os) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioC\ n\,) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioA\ n \mathbin{;} R & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet & [\text{one point}] \\
\qquad (\exists\, c : LocalC \bullet & \\
\qquad\qquad a \mapsto c \in r & \\
\qquad\qquad \land & \\
\qquad\qquad (c, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioC\ n\,) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (a, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioA\ n \mathbin{;} R & \\
\quad \land & \\
\quad (r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \bullet \qquad [\mathit{batch}^T] \\
\qquad (\exists\, c : \mathit{LocalC} \bullet \\
\qquad\qquad a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{ins}', \mathit{outs}') \in \mathit{batch}\ \mathit{ioOps}\ \mathit{ioC}\ n\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \operatorname{;} R \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \bullet \qquad [\mathit{batch}] \\
\qquad (\exists\, c : \mathit{LocalC} \bullet \\
\qquad\qquad a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{head}\ \mathit{ins}) \mapsto (c', \mathit{last}\ \mathit{outs}') \in \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad\qquad \wedge \\
\qquad\qquad \mathit{ins}' = \mathit{tail}\ \mathit{ins}\ \wedge\ \mathit{outs} = \mathit{first}\ \mathit{outs}'\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \operatorname{;} R \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \qquad [\mathrm{def}^n \text{ of } \Rightarrow] \\
\qquad\qquad\qquad \mathit{ins}' = \mathit{tail}\ \mathit{ins}\ \wedge\ \mathit{outs} = \mathit{first}\ \mathit{outs}' \bullet \\
\qquad (\exists\, c : \mathit{LocalC} \bullet \\
\qquad\qquad a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{head}\ \mathit{ins}) \mapsto (c', \mathit{last}\ \mathit{outs}') \in \mathit{ioOps}\ \mathit{ioC}\ n\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \operatorname{;} R \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{in} : \mathit{Value}; \qquad [\mathrm{def}^n \text{ of } \mathit{tail}] \\
\qquad\qquad\qquad\qquad \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad\qquad\qquad \mathit{ins} = \langle \mathit{in} \rangle \frown \mathit{ins}'\ \wedge\ \mathit{outs} = \mathit{first}\ \mathit{outs}' \bullet \\
\qquad (\exists\, c : \mathit{LocalC} \bullet \\
\qquad\qquad a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \mathit{head}\ \mathit{ins}) \mapsto (c', \mathit{last}\ \mathit{outs}') \in \mathit{ioOps}\ \mathit{ioC}\ n\ ) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (a, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \operatorname{;} R \\
\quad \wedge \\
\quad (r \parallel \mathrm{id})(\!|\ \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \qquad [\mathrm{def}^n \text{ of } \mathit{first}]\\
\qquad ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad ins = \langle in \rangle \frown ins' \ \wedge\ outs' = outs \frown \langle out \rangle \bullet \\
\quad (\exists\, c : LocalC \bullet \\
\qquad a \mapsto c \in r \\
\qquad \wedge \\
\qquad (c, head\ ins) \mapsto (c', last\ outs') \in ioOps\ ioC\ n\,) \\
\qquad \Rightarrow \\
\qquad (a, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioA\ n \fatsemi R \\
\wedge \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \qquad [\text{one point rule}]\\
\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\quad (\exists\, c : LocalC \bullet a \mapsto c \in r \\
\qquad\qquad \wedge \\
\qquad\qquad (c, in) \mapsto (c', outs) \in ioOps\ ioC\ n\,) \\
\qquad \Rightarrow \\
\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (c', ins', outs \frown \langle out \rangle) \\
\qquad\quad \in \\
\qquad\quad batch^T\ ioOps\ ioA\ n \fatsemi R \\
\wedge \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \qquad [\text{property of id}]\\
\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\quad (\exists\, c : LocalC \bullet (a, in) \mapsto (c, in) \in (r \parallel \mathrm{id}) \\
\qquad\qquad \wedge \\
\qquad\qquad (c, in) \mapsto (c', outs) \in ioOps\ ioC\ n\,) \\
\qquad \Rightarrow \\
\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (c', ins', outs \frown \langle out \rangle) \\
\qquad\quad \in \\
\qquad\quad batch^T\ ioOps\ ioA\ n \fatsemi R \\
\wedge \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \qquad [\mathrm{def}^n \text{ of } \fatsemi]\\
\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\quad (a, in) \mapsto (c', outs) \in (r \parallel \mathrm{id}) \fatsemi ioOps\ ioC\ n \\
\qquad \Rightarrow \\
\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (c', ins', outs \frown \langle out \rangle) \\
\qquad\quad \in \\
\qquad\quad batch^T\ ioOps\ ioA\ n \fatsemi R \\
\wedge \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \hfill [\mathrm{def}^n \text{ of } \mathbin{;}]\\
\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad (a, in) \mapsto (c', outs) \in (r \parallel \mathrm{id}) \mathbin{;} ioOps\ ioC\ n\\
\qquad\quad \Rightarrow\\
\qquad\quad (\exists\, a' : LocalA;\ is, os : \mathrm{seq}\ Value \bullet\\
\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', is, os) \in batch^T\ ioOps\ ioA\ n\\
\qquad\qquad \wedge\\
\qquad\qquad (a', is, os) \mapsto (c', ins', outs \frown \langle out \rangle) \in R\\
\wedge\\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \hfill [\mathrm{def}^n \text{ of } R]\\
\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad (a, in) \mapsto (c', outs) \in (r \parallel \mathrm{id}) \mathbin{;} ioOps\ ioC\ n\\
\qquad\quad \Rightarrow\\
\qquad\quad (\exists\, a' : LocalA;\ is, os : \mathrm{seq}\ Value \bullet\\
\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', is, os) \in batch^T\ ioOps\ ioA\ n\\
\qquad\qquad \wedge\\
\qquad\qquad a' \mapsto c' \in r\ \wedge\ is = ins'\ \wedge\ os = outs \frown \langle out \rangle\\
\wedge\\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ c' : LocalC;\ in, out : Value; \hfill [\text{one point rule}]\\
\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet\\
\qquad (a, in) \mapsto (c', outs) \in (r \parallel \mathrm{id}) \mathbin{;} ioOps\ ioC\ n\\
\qquad\quad \Rightarrow\\
\qquad\quad (\exists\, a' : LocalA \bullet\\
\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', ins', outs \frown \langle out \rangle)\\
\qquad\qquad\quad \in\\
\qquad\qquad\quad batch^T\ ioOps\ ioA\ n\\
\qquad\qquad \wedge\\
\qquad\qquad a' \mapsto c' \in r\\
\wedge\\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ ioOps\ ioA\ n\ |\!) \subseteq \mathrm{dom}\ ioOps\ ioC\ n
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{in}, \mathit{out} : \mathit{Value}; & [\mathrm{def}^{n} \text{ of } \mathit{batch}^{T}] \\
\qquad \mathit{outs}, \mathit{ins}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (a, \mathit{in}) \mapsto (c', \mathit{outs}) \in (r \parallel \mathrm{id}) \mathbin{⨾} \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a' : \mathit{LocalA} \bullet & \\
\qquad\quad (a, \langle \mathit{in} \rangle \frown \mathit{ins}', \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs} \frown \langle \mathit{out} \rangle) & \\
\qquad\qquad \in & \\
\qquad\qquad \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\qquad\quad \wedge & \\
\qquad\quad a' \mapsto c' \in r & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{in}, \mathit{out} : \mathit{Value} \bullet & [\mathrm{def}^{n} \text{ of } \mathit{batch}] \\
\quad (a, \mathit{in}) \mapsto (c', \mathit{outs}) \in (r \parallel \mathrm{id}) \mathbin{⨾} \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a' : \mathit{LocalA} \bullet & \\
\qquad\quad (a, \mathit{in}) \mapsto (a', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioA}\ n & \\
\qquad\quad \wedge & \\
\qquad\quad a' \mapsto c' \in r & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{in}, \mathit{out} : \mathit{Value} \bullet & [\text{property of id}] \\
\quad (a, \mathit{in}) \mapsto (c', \mathit{outs}) \in (r \parallel \mathrm{id}) \mathbin{⨾} \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a' : \mathit{LocalA} \bullet & \\
\qquad\quad (a, \mathit{in}) \mapsto (a', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioA}\ n & \\
\qquad\quad \wedge & \\
\qquad\quad (a', \mathit{out}) \mapsto (c', \mathit{out}) \in (r \parallel \mathrm{id}) & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : \mathit{LocalA};\ c' : \mathit{LocalC};\ \mathit{in}, \mathit{out} : \mathit{Value} \bullet & [\mathrm{def}^{n} \text{ of } ⨾] \\
\quad (a, \mathit{in}) \mapsto (c', \mathit{outs}) \in (r \parallel \mathrm{id}) \mathbin{⨾} \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \Rightarrow & \\
\qquad (a, \mathit{in}) \mapsto (c', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{⨾} (r \parallel \mathrm{id}) & \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (r \parallel \mathrm{id}) \mathbin{⨾} \mathit{ioOps}\ \mathit{ioC}\ n \subseteq \mathit{ioOps}\ \mathit{ioA}\ n \mathbin{⨾} (r \parallel \mathrm{id}) & [\mathrm{def}^{n} \text{ of } \subseteq] \\
\wedge & \\
(r \parallel \mathrm{id})(\!|\ \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\ |\!) \subseteq \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n &
\end{array}
$$

A.2 Backwards simulations rules

In this section we prove that the relaxed and totalised versions of the backwards simulation rules for ioData types are equivalent

Theorem 7 Let $ioA : ioData[LocalA, Name, Value]$ and $ioC : ioData[LocalC, Name, Value]$ be ioData types such that there exist relations s and S as defined below.

$$s : LocalC \leftrightarrow LocalA$$

and

$$\begin{array}{l} S : (\, LocalC \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \leftrightarrow \\ \qquad (\, LocalA \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value\,) \\ \hline S = \{a : LocalA;\ c : LocalC;\ ins, outs : \mathrm{seq}\ Value \mid \\ \qquad c \mapsto a \in s \bullet (c, ins, outs) \mapsto (a, ins, outs)\} \end{array}$$

Then Rules A.10, A.11 and A.12 below

$$startBatch^T\ ioC \mathbin{;} S^T \ \subseteq\ startBatch^T\ ioA \tag{A.10}$$

$$stopBatch^T\ ioC \ \subseteq\ S^T \mathbin{;} stopBatch^T\ ioA \tag{A.11}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\, ioOps\ ioC \bullet \\ \qquad batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \ \subseteq\ S^T \mathbin{;} batch^T\ ioOps\ ioA\ n \end{array} \tag{A.12}$$

hold precisely when Rules A.13, A.14, A.15, A.16, A.17 and A.18 hold.

$$ioInit\ ioC \mathbin{;} (s \parallel \mathrm{id}) \ \subseteq\ ioInit\ ioA \tag{A.13}$$

$$ioFinal\ ioC \ \subseteq\ (s \parallel \mathrm{id}) \mathbin{;} ioFinal\ ioA \tag{A.14}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\, ioOps\ ioC \bullet \\ \qquad ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \ \subseteq\ (s \parallel \mathrm{id}) \mathbin{;} ioOps\ ioA\ n \end{array} \tag{A.15}$$

$$\mathrm{dom}\, ioInit\ ioA \ \subseteq\ \mathrm{dom}\, ioInit\ ioC \tag{A.16}$$

$$\begin{array}{l} (LocalC \times Value) \setminus \mathrm{dom}\, ioFinal\ ioC \\ \qquad \subseteq \\ \qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\, ioFinal\ ioA\,) \end{array} \tag{A.17}$$

$$\begin{array}{l} \forall\, n : \mathrm{dom}\, ioOps\ ioC \bullet \\ \qquad (LocalC \times Value) \setminus \mathrm{dom}\, ioOps\ ioC\ n \\ \qquad\qquad \subseteq \\ \qquad\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\, ioOps\ ioA\ n) \end{array} \tag{A.18}$$

□

Proof

By Lemmas 4, 5 and 6.

Lemma 4

$$\begin{aligned}
&startBatch^{T}\ ioC \circ S^{T} \rhd BatchState[LocalA, Value] \\
&\quad \Leftrightarrow (\ ioInit\ ioC \circ (s \parallel \mathrm{id}) \ \subseteq\ ioInit\ ioA \\
&\qquad \wedge \\
&\qquad \mathrm{dom}\ ioInit\ ioA = \mathrm{dom}\ ioInit\ ioC
\end{aligned}$$

□

$$\begin{aligned}
&startBatch^{T}\ ioC \circ S^{T} \ \subseteq\ startBatch^{T}\ ioA \\
&\quad \Leftrightarrow startBatch^{T}\ ioC \circ S^{T} \rhd BatchState[LocalA, Value] && \text{[property of } \subseteq] \\
&\qquad\quad \subseteq \\
&\qquad\quad startBatch^{T}\ ioA \rhd BatchState[LocalA, Value] \\
&\qquad \wedge \\
&\qquad startBatch^{T}\ ioC \circ S^{T} \rhd BatchState[\{\perp\}, Value] \\
&\qquad\quad \subseteq \\
&\qquad\quad startBatch^{T}\ ioA \rhd BatchState[\{\perp\}, Value] \\
&\quad \Leftrightarrow startBatch^{T}\ ioC \circ S^{T} \rhd BatchState[LocalA, Value] && [\mathrm{def}^{n} \text{ of } \subseteq] \\
&\qquad\quad \subseteq \\
&\qquad\quad startBatch^{T}\ ioA \rhd BatchState[LocalA, Value] \\
&\qquad \wedge \\
&\qquad \forall\, is, is', os' : \mathrm{seq}\ Value \bullet \\
&\qquad\quad is \mapsto (\perp, is', os') \in startBatch^{T}\ ioC \circ S^{T} \\
&\qquad\qquad \Rightarrow \\
&\qquad\qquad is \mapsto (\perp, is', os') \in startBatch^{T}\ ioA \\
&\quad \Leftrightarrow startBatch^{T}\ ioC \circ S^{T} \rhd BatchState[LocalA, Value] && [\mathrm{def}^{n} \text{ of } \circ] \\
&\qquad\quad \subseteq \\
&\qquad\quad startBatch^{T}\ ioA \rhd BatchState[LocalA, Value] \\
&\qquad \wedge \\
&\qquad \forall\, is, is', os' : \mathrm{seq}\ Value \bullet \\
&\qquad\quad \exists\, c : LocalC_{\perp};\ is'', os'' : \mathrm{seq}\ Value \bullet \\
&\qquad\qquad\quad is \mapsto (c, is'', os'') \in startBatch^{T}\ ioC \\
&\qquad\qquad\quad \wedge \\
&\qquad\qquad\quad (c, is'', os'') \mapsto (\perp, is', os') \in S^{T} \\
&\qquad\qquad \Rightarrow \\
&\qquad\qquad is \mapsto (\perp, is', os') \in startBatch^{T}\ ioA
\end{aligned}$$

$$
\begin{array}{l}
\Leftrightarrow startBatch^T\ ioC \fatsemi S^T \rhd BatchState[LocalA, Value] \qquad [\mathrm{def}^n \text{ of } S^T]\\
\qquad \subseteq\\
\qquad startBatch^T\ ioA \rhd BatchState[LocalA, Value]\\
\quad \wedge\\
\quad \forall\, is, is', os' : \mathrm{seq}\ Value \bullet\\
\qquad \exists\, c : LocalC_\bot;\ is'', os'' : \mathrm{seq}\ Value \bullet\\
\qquad\qquad is \mapsto (c, is'', os'') \in startBatch^T\ ioC\\
\qquad\qquad \wedge\\
\qquad\qquad c = \bot\ \wedge\ is'' = is'\ \wedge\ os'' = os'\\
\qquad\quad \Rightarrow\\
\qquad\quad is \mapsto (\bot, is', os') \in startBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow startBatch^T\ ioC \fatsemi S^T \rhd BatchState[LocalA, Value] \qquad [\text{one point rule}]\\
\qquad \subseteq\\
\qquad startBatch^T\ ioA \rhd BatchState[LocalA, Value]\\
\quad \wedge\\
\quad \forall\, is, is', os' : \mathrm{seq}\ Value \bullet\\
\qquad is \mapsto (\bot, is', os') \in startBatch^T\ ioC\\
\qquad\quad \Rightarrow\\
\qquad\quad is \mapsto (\bot, is', os') \in startBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow startBatch^T\ ioC \fatsemi S^T \rhd BatchState[LocalA, Value] \qquad [startBatch^T]\\
\qquad \subseteq\\
\qquad startBatch^T\ ioA \rhd BatchState[LocalA, Value]\\
\quad \wedge\\
\quad \forall\, is, is', os' : \mathrm{seq}\ Value \bullet\\
\qquad head\ is \notin \mathrm{dom}\ ioInit\ ioC\ \wedge\ is' = tail\ is \wedge\ os' = \langle\rangle\\
\qquad\quad \Rightarrow\\
\qquad\quad is \mapsto (\bot, is', os') \in startBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow startBatch^T\ ioC \fatsemi S^T \rhd BatchState[LocalA, Value] \qquad [\text{one point rule}]\\
\qquad \subseteq\\
\qquad startBatch^T\ ioA \rhd BatchState[LocalA, Value]\\
\quad \wedge\\
\quad \forall\, is, is' : \mathrm{seq}\ Value \bullet\\
\qquad head\ is \notin \mathrm{dom}\ ioInit\ ioC\ \wedge\ is' = tail\ is\\
\qquad\quad \Rightarrow\\
\qquad\quad is \mapsto (\bot, is', \langle\rangle\ ) \in startBatch^T\ ioA
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \,{}_{9}^{\circ}\, S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \mathit{tail}] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \forall\, \mathit{in} : \mathit{Value};\ \mathit{is}, \mathit{is}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad \mathit{head}\ \mathit{is} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \ \wedge\ \mathit{is} = \langle \mathit{in} \rangle \frown \mathit{is}' & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \mathit{is} \mapsto (\bot, \mathit{is}',\ \langle\rangle\ ) \in \mathit{startBatch}^T\ \mathit{ioA} & \\
\\
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \,{}_{9}^{\circ}\, S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{one point rule}] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \forall\, \mathit{in} : \mathit{Value};\ \mathit{is}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad \mathit{in} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \langle \mathit{in} \rangle \frown \mathit{is}' \mapsto (\bot, \mathit{is}',\ \langle\rangle\ ) \in \mathit{startBatch}^T\ \mathit{ioA} & \\
\\
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \,{}_{9}^{\circ}\, S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\mathit{startBatch}^T] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \forall\, \mathit{in} : \mathit{Value} \bullet & \\
\qquad \mathit{in} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \mathit{in} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} & \\
\\
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \,{}_{9}^{\circ}\, S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\text{property of } \Rightarrow] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \forall\, \mathit{in} : \mathit{Value} \bullet & \\
\qquad \mathit{in} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \mathit{in} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} & \\
\\
\Leftrightarrow \mathit{startBatch}^T\ \mathit{ioC} \,{}_{9}^{\circ}\, S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^T\ \mathit{ioA} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \subseteq \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of } \subseteq] \\
\quad ins \mapsto (a, ins', outs') \in startBatch^{T}\ ioC \mathbin{⨾} S^{T} & \\
\qquad \Rightarrow & \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of } ⨾] \\
\qquad \exists\, c : LocalC;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad ins \mapsto (c, is, os) \in startBatch^{T}\ ioC & \\
\qquad\qquad \wedge & \\
\qquad\qquad (c, is, os) \mapsto (a, ins', outs') \in S^{T} & \\
\qquad \Rightarrow & \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of } S^{T}] \\
\qquad \exists\, c : LocalC;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad ins \mapsto (c, is, os) \in startBatch^{T}\ ioC & \\
\qquad\qquad \wedge & \\
\qquad\qquad c \mapsto a \in s\ \wedge\ is = ins'\ \wedge\ os = outs' & \\
\qquad \Rightarrow & \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet & [\text{one point rule}] \\
\qquad \exists\, c : LocalC \bullet & \\
\qquad\qquad ins \mapsto (c, ins', outs') \in startBatch^{T}\ ioC & \\
\qquad\qquad \wedge & \\
\qquad\qquad c \mapsto a \in s & \\
\qquad \Rightarrow & \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet \qquad [\mathrm{def}^n \text{ of } startBatch^T] \\
\quad \exists\, c : LocalC \bullet \\
\qquad\quad head\ ins \mapsto (c, last\ outs') \in ioInit\ ioC \\
\qquad\quad \wedge \\
\qquad\quad ins' = tail\ ins\ \wedge\ \#outs' = 1 \\
\qquad\quad \wedge \\
\qquad\quad c \mapsto a \in s \\
\qquad \Rightarrow \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^T\ ioA \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ ins, ins', outs' : \mathrm{seq}\ Value\ | \qquad [\text{property of } \Rightarrow] \\
\quad ins' = tail\ ins\ \wedge\ \#outs' = 1 \bullet \\
\qquad \exists\, c : LocalC \bullet \\
\qquad\quad head\ ins \mapsto (c, last\ outs') \in ioInit\ ioC \\
\qquad\quad \wedge \\
\qquad\quad c \mapsto a \in s \\
\qquad \Rightarrow \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^T\ ioA \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ out : Value;\ ins, ins', outs' : \mathrm{seq}\ Value\ | \qquad [\mathrm{def}^n \text{ of } \#] \\
\quad ins' = tail\ ins\ \wedge\ \#outs' = \langle out \rangle \bullet \\
\qquad \exists\, c : LocalC \bullet \\
\qquad\quad head\ ins \mapsto (c, last\ outs') \in ioInit\ ioC \\
\qquad\quad \wedge \\
\qquad\quad c \mapsto a \in s \\
\qquad \Rightarrow \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^T\ ioA \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, a : LocalA;\ in, out : Value;\ ins, ins', outs' : \mathrm{seq}\ Value\ | \qquad [tail] \\
\quad ins = \langle in \rangle \frown ins'\ \wedge\ \#outs' = \langle out \rangle \bullet \\
\qquad \exists\, c : LocalC \bullet \\
\qquad\quad head\ ins \mapsto (c, last\ outs') \in ioInit\ ioC \\
\qquad\quad \wedge \\
\qquad\quad c \mapsto a \in s \\
\qquad \Rightarrow \\
\qquad ins \mapsto (a, ins', outs') \in startBatch^T\ ioA \\
\wedge \\
\mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ in, out : Value;\ ins' : \mathrm{seq}\ Value \bullet & \text{[one point rule]} \\
\qquad \exists\, c : LocalC \bullet in \mapsto (c, out) \in ioInit\ ioC & \\
\qquad\qquad \wedge & \\
\qquad\qquad c \mapsto a \in s & \\
\qquad \Rightarrow & \\
\qquad \langle in \rangle \frown ins' \mapsto (a, ins', \langle out \rangle) \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ in, out : Value;\ ins' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of id}] \\
\qquad \exists\, c : LocalC \bullet in \mapsto (c, out) \in ioInit\ ioC & \\
\qquad\qquad \wedge & \\
\qquad\qquad (c, out) \mapsto (a, out) \in (s \parallel \mathrm{id}) & \\
\qquad \Rightarrow & \\
\qquad \langle in \rangle \frown ins' \mapsto (a, ins', \langle out \rangle) \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ in, out : Value;\ ins' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of } \fatsemi] \\
\qquad in \mapsto (a, out) \in ioInit\ ioC \fatsemi (s \parallel \mathrm{id}) & \\
\qquad \Rightarrow & \\
\qquad \langle in \rangle \frown ins' \mapsto (a, ins', \langle out \rangle) \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, a : LocalA;\ in, out : Value;\ ins' : \mathrm{seq}\ Value \bullet & [startBatch^{T}] \\
\qquad in \mapsto (a, out) \in ioInit\ ioC \fatsemi (s \parallel \mathrm{id}) & \\
\qquad \Rightarrow & \\
\qquad in \mapsto (a, out) \in ioInit\ ioA & \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow ioInit\ ioC \fatsemi (s \parallel \mathrm{id}) \subseteq ioInit\ ioA & [\mathrm{def}^{n} \text{ of } \subseteq] \\
\quad \wedge & \\
\quad \mathrm{dom}\ ioInit\ ioA \subseteq \mathrm{dom}\ ioInit\ ioC &
\end{array}
$$

Lemma 5

$$
\begin{array}{l}
stopBatch^{T}\ ioC \subseteq S^{T} \fatsemi stopBatch^{T}\ ioA \\
\quad \Leftrightarrow (\ ioFinal\ ioC \subseteq (s \parallel \mathrm{id}) \fatsemi ioFinal\ ioA \\
\qquad \wedge \\
\qquad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\ )
\end{array}
$$

□

$$
\begin{array}{ll}
stopBatch^T\ ioC \subseteq S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [\text{property of } \subseteq] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad BatchState[\{\perp\}, Value] \lhd stopBatch^T\ ioC & \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[\{\perp\}, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [stopBatch^T] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\} & \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[\{\perp\}, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [\mathrm{def}^n \text{ of } S^T] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\} & \\
\qquad\quad \subseteq & \\
\qquad\quad \mathrm{id}[BatchState[\{\perp\}, Value]] \mathbin{⨾} stopBatch^T\ ioA & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [\text{property of id}] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\} & \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[\{\perp\}, Value] \lhd stopBatch^T\ ioA & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [stopBatch^T] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\} & \\
\qquad\quad \subseteq & \\
\qquad\quad \{in : Value;\ outs : \mathrm{seq}\ Value \bullet (\perp, \langle in \rangle, outs) \mapsto outs\} & \\
\quad \Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC & [\text{property of } \subseteq] \\
\qquad\quad \subseteq & \\
\qquad\quad BatchState[LocalC, Value] \lhd S^T \mathbin{⨾} stopBatch^T\ ioA & \\
\qquad \wedge & \\
\qquad \mathrm{true} &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & [\text{property of } \wedge] \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} & \\[1em]
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & [\text{def}^n \text{ of } S^T] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} & \\[1em]
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{prop. of } \subseteq] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\ \wedge & \\
\ \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \cap \mathit{third} & \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} & \\[1em]
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{def}^n \text{ of } \subseteq] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\ \wedge & \\
\ \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os}, \mathit{os}' : \operatorname{seq} \mathit{Value} \bullet & \\
\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' & \\
\qquad\qquad\qquad \in & \\
\qquad\qquad\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \cap \mathit{third} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} & \\[1em]
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{prop. of } \in] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\ \wedge & \\
\ \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os}, \mathit{os}' : \operatorname{seq} \mathit{Value} \bullet & \\
\qquad (\,(c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & \\
\qquad\ \ \wedge & \\
\qquad\ \ (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{third}\,) & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} &
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\mathrm{def}^n \text{ of } \mathit{third}] \\
\qquad \subseteq & \\
\qquad S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os}, \mathit{os}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad ((c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & \\
\qquad\ \land & \\
\qquad\ \mathit{os}' = \mathit{os} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{prop. of} \Rightarrow] \\
\qquad \subseteq & \\
\qquad S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os}, \mathit{os}' : \mathrm{seq}\ \mathit{Value} \mid \mathit{os}' = \mathit{os} \bullet & \\
\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os}' \in S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{one point rule}] \\
\qquad \subseteq & \\
\qquad S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\mathit{stopBatch}^T] \\
\qquad \subseteq & \\
\qquad S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in \{\mathit{local} : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{outs} : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad\qquad (\mathit{local}, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \bullet & \\
\qquad\qquad\qquad\qquad\quad (\mathit{local}, \langle \mathit{in} \rangle, \mathit{outs}) \mapsto \mathit{outs}\} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in S \mathbin{⨾} \mathit{stopBatch}^T\ \mathit{ioA} \cap \mathit{third} &
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC} \setminus \mathit{third} \qquad \text{[set theory]} \\
\qquad \subseteq \\
\qquad S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\quad \wedge \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet \\
\qquad (c, \mathit{head}\ \mathit{is}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC}\ \wedge\ \#\mathit{is} = 1 \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \cap \mathit{third} \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC} \setminus \mathit{third} \qquad \text{[property of } \Rightarrow\text{]} \\
\qquad \subseteq \\
\qquad S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\quad \wedge \\
\quad \forall c : \mathit{LocalC};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{is} = 1 \bullet \\
\qquad (c, \mathit{head}\ \mathit{is}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \cap \mathit{third} \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC} \setminus \mathit{third} \qquad \text{[def}^{n} \text{ of } \#] \\
\qquad \subseteq \\
\qquad S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\quad \wedge \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \mid \mathit{is} = \langle \mathit{in} \rangle \bullet \\
\qquad (c, \mathit{head}\ \mathit{is}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \mathit{is}, \mathit{os}) \mapsto \mathit{os} \in S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \cap \mathit{third} \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC} \setminus \mathit{third} \qquad \text{[one point rule]} \\
\qquad \subseteq \\
\qquad S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\quad \wedge \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \langle \mathit{in} \rangle, \mathit{os}) \mapsto \mathit{os} \in S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \cap \mathit{third} \\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^{T}\ \mathit{ioC} \setminus \mathit{third} \qquad \text{[prop. of } \cap] \\
\qquad \subseteq \\
\qquad S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\quad \wedge \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\quad \Rightarrow \\
\qquad\quad (\,(c, \langle \mathit{in} \rangle, \mathit{os}) \mapsto \mathit{os} \in S \fatsemi \mathit{stopBatch}^{T}\ \mathit{ioA} \\
\qquad\qquad \wedge \\
\qquad\qquad (c, \langle \mathit{in} \rangle, \mathit{os}) \mapsto \mathit{os} \in \mathit{third}\,)
\end{array}
$$

$$\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC \setminus third \qquad [\text{def}^n \text{ of } third] \\
\qquad \subseteq \\
\qquad S \mathbin{;} stopBatch^T\ ioA \setminus third \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value;\ os : \text{seq } Value \bullet \\
\qquad (c, in) \notin \text{dom } ioFinal\ ioC \\
\qquad\quad \Rightarrow \\
\qquad\quad (\,(c, \langle in \rangle, os) \mapsto os \in S \mathbin{;} stopBatch^T\ ioA \\
\qquad\quad\ \ \wedge \\
\qquad\quad\ \ \text{true}\,)
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC \setminus third \qquad [\text{prop. of } \wedge] \\
\qquad \subseteq \\
\qquad S \mathbin{;} stopBatch^T\ ioA \setminus third \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value;\ os : \text{seq } Value \bullet \\
\qquad (c, in) \notin \text{dom } ioFinal\ ioC \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \langle in \rangle, os) \mapsto os \in S \mathbin{;} stopBatch^T\ ioA
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^T\ ioC \setminus third \qquad [\text{def}^n \text{ of } \mathbin{;}] \\
\qquad \subseteq \\
\qquad S \mathbin{;} stopBatch^T\ ioA \setminus third \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value;\ os : \text{seq } Value \bullet \\
\qquad (c, in) \notin \text{dom } ioFinal\ ioC \\
\qquad\quad \Rightarrow \\
\qquad\quad \exists a : LocalA;\ ins, outs : \text{seq } Value \bullet \\
\qquad\qquad (\,(c, \langle in \rangle, os) \mapsto (a, ins, outs) \in S \\
\qquad\qquad\ \ \wedge \\
\qquad\qquad\ \ (a, ins, outs) \mapsto os \\
\qquad\qquad\qquad \in \\
\qquad\qquad\qquad BatchState[LocalA, Value] \lhd stopBatch^T\ ioA\,)
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\mathrm{def}^n \text{ of } S] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists\, a : \mathit{LocalA};\ \mathit{ins}, \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad (\, c \mapsto a \in s \ \land\ \mathit{ins} = \langle \mathit{in} \rangle \ \land\ \mathit{outs} = \mathit{os} & \\
\qquad\qquad\ \land & \\
\qquad\qquad\ (a, \mathit{ins}, \mathit{outs}) \mapsto \mathit{os} & \\
\qquad\qquad\qquad \in & \\
\qquad\qquad\qquad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) & \\
\\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{one point rule}] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists\, a : \mathit{LocalA};\ \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad (\, c \mapsto a \in s & \\
\qquad\qquad\ \land & \\
\qquad\qquad\ (a, \langle \mathit{in} \rangle, \mathit{os}) \mapsto \mathit{os} & \\
\qquad\qquad\qquad \in & \\
\qquad\qquad\qquad \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioA}\,) & \\
\\
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\mathit{stopBatch}^T] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists\, a : \mathit{LocalA};\ \mathit{outs} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad (\, c \mapsto a \in s & \\
\qquad\qquad\ \land & \\
\qquad\qquad\ (a, \langle \mathit{in} \rangle, \mathit{os}) \mapsto \mathit{os} & \\
\qquad\qquad\qquad \in & \\
\qquad\qquad\qquad \{ \mathit{local} : \mathit{LocalA};\ \mathit{in} : \mathit{Value};\ \mathit{outs} : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad\quad (\mathit{local}, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet & \\
\qquad\qquad\qquad\qquad (\mathit{local}, \langle \mathit{in} \rangle, \mathit{outs}) \mapsto \mathit{outs} \}\,) &
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{set theory}] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists a : \mathit{LocalA} \bullet & \\
\qquad\qquad (\ c \mapsto a \in s & \\
\qquad\qquad\ \ \land & \\
\qquad\qquad\ \ (a, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}\ ) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{def}^n \text{ if id}] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists a : \mathit{LocalA} \bullet & \\
\qquad\qquad (\ (c, \mathit{in}) \mapsto (a, \mathit{in}) \in (s \parallel \mathrm{id}) & \\
\qquad\qquad\ \ \land & \\
\qquad\qquad\ \ (a, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}\ ) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{def}^n \text{ of } \rhd] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists a : \mathit{LocalA} \bullet & \\
\qquad\qquad (c, \mathit{in}) \mapsto (a, \mathit{in}) \in (s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd \mathit{stopBatch}^T\ \mathit{ioC} \setminus \mathit{third} & [\text{def}^n \text{ of dom}] \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA} \setminus \mathit{third} & \\
\quad \land & \\
\quad \forall c : \mathit{LocalC};\ \mathit{in} : \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, \mathit{in}) \in \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}\ ) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^{T}\ ioC \setminus third & [\text{def}^{\,n} \text{ of } \setminus] \\
\qquad \subseteq & \\
\qquad S \fatsemi stopBatch^{T}\ ioA \setminus third & \\
\quad \land & \\
\quad \forall\, c : LocalC;\ in : Value \bullet & \\
\qquad (c, in) \in (LocalC \times Value) \setminus \operatorname{dom} ioFinal\ ioC & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, in) \in \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioFinal\ ioA\,) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow BatchState[LocalC, Value] \lhd stopBatch^{T}\ ioC \setminus third & [\text{def}^{\,n} \text{ of } \subseteq] \\
\qquad \subseteq & \\
\qquad S \fatsemi stopBatch^{T}\ ioA \setminus third & \\
\quad \land & \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioFinal\ ioA\,) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \operatorname{seq} Value \bullet & [\text{def}^{\,n} \text{ of } \subseteq] \\
\qquad (c, ins, outs) \mapsto outs' \in stopBatch^{T}\ ioC \setminus third & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, ins, outs) \mapsto outs' \in S \fatsemi stopBatch^{T}\ ioA \setminus third & \\
\quad \land & \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioFinal\ ioA\,) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \operatorname{seq} Value \bullet & [\text{def}^{\,n} \text{ of } \setminus] \\
\qquad (\,(c, ins, outs) \mapsto outs' \in stopBatch^{T}\ ioC & \\
\qquad\ \ \land & \\
\qquad\ \ (c, ins, outs) \mapsto outs' \notin third\,) & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (c, ins, outs) \mapsto outs' \in S \fatsemi stopBatch^{T}\ ioA \setminus third & \\
\quad \land & \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioFinal\ ioA\,) &
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } third] \\
\quad ((c, ins, outs) \mapsto outs' \in stopBatch^T\ ioC & \\
\quad\ \wedge & \\
\quad\ outs' \neq outs\ ) & \\
\qquad \Rightarrow & \\
\qquad (c, ins, outs) \mapsto outs' \in S \mathbin{;} stopBatch^T\ ioA \setminus third & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\ ) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \mathrm{seq}\ Value \mid & [\text{property of } \Rightarrow] \\
\qquad\qquad outs' \neq outs \bullet & \\
\quad (c, ins, outs) \mapsto outs' \in stopBatch^T\ ioC & \\
\qquad \Rightarrow & \\
\qquad (c, ins, outs) \mapsto outs' \in S \mathbin{;} stopBatch^T\ ioA \setminus third & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\ ) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \mathrm{seq}\ Value \mid & [\mathrm{def}^n \text{ of } stopBatch^T] \\
\qquad\qquad outs' \neq outs \bullet & \\
\quad (c, ins, outs) \mapsto outs' \in stopBatch\ ioC & \\
\qquad \Rightarrow & \\
\qquad (c, ins, outs) \mapsto outs' \in S \mathbin{;} stopBatch^T\ ioA \setminus third & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\ ) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ ins, outs, outs' : \mathrm{seq}\ Value \mid & [\mathrm{def}^n \text{ of } stopBatch] \\
\qquad\qquad outs' \neq outs \bullet & \\
\quad (c, ins, outs) \mapsto outs' & \\
\qquad\qquad \in & \\
\qquad\qquad \{inF, outF : Value;\ outs : \mathrm{seq}\ Value;\ local : LocalC \mid & \\
\qquad\qquad\quad (local, inF) \mapsto outF \in ioFinal\ ioC \bullet & \\
\qquad\qquad\qquad (local, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle\} & \\
\qquad \Rightarrow & \\
\qquad (c, ins, outs) \mapsto outs' \in S \mathbin{;} stopBatch^T\ ioA \setminus third & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\ ) &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \qquad [\text{set theory}] \\
\qquad\qquad \mathit{outs}' \neq \mathit{outs} \bullet \\
\qquad ((c, \mathit{head\ ins}) \mapsto \mathit{last\ outs}' \in \mathit{ioFinal\ ioC} \\
\qquad\ \wedge \\
\qquad\ \#\mathit{ins} = 1\ \wedge\ \mathit{outs} = \mathit{first\ outs}'\,) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{outs}' \in S \mathbin{;} \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\ \wedge \\
\ (\mathit{LocalC} \times \mathit{Value}) \setminus \operatorname{dom} \mathit{ioFinal\ ioC} \\
\qquad \subseteq \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} \mathit{ioFinal\ ioA}\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \qquad [\text{property of} \Rightarrow] \\
\qquad\qquad \#\mathit{ins} = 1\ \wedge\ \mathit{outs} = \mathit{first\ outs}' \bullet \\
\qquad (c, \mathit{head\ ins}) \mapsto \mathit{last\ outs}' \in \mathit{ioFinal\ ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{outs}' \in S \mathbin{;} \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\ \wedge \\
\ (\mathit{LocalC} \times \mathit{Value}) \setminus \operatorname{dom} \mathit{ioFinal\ ioC} \\
\qquad \subseteq \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} \mathit{ioFinal\ ioA}\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \qquad [\text{def}^{\,n} \text{ of } \#] \\
\qquad\qquad \mathit{ins} = \langle \mathit{inF} \rangle\ \wedge\ \mathit{outs} = \mathit{first\ outs}' \bullet \\
\qquad (c, \mathit{head\ ins}) \mapsto \mathit{last\ outs}' \in \mathit{ioFinal\ ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{outs}' \in S \mathbin{;} \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\ \wedge \\
\ (\mathit{LocalC} \times \mathit{Value}) \setminus \operatorname{dom} \mathit{ioFinal\ ioC} \\
\qquad \subseteq \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} \mathit{ioFinal\ ioA}\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ \mathit{inF}, \mathit{outF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \qquad [\mathit{first}] \\
\qquad\qquad \mathit{ins} = \langle \mathit{inF} \rangle\ \wedge\ \mathit{outs}' = \mathit{outs} \frown \langle \mathit{outF} \rangle \bullet \\
\qquad (c, \mathit{head\ ins}) \mapsto \mathit{last\ outs}' \in \mathit{ioFinal\ ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto \mathit{outs}' \in S \mathbin{;} \mathit{stopBatch}^{T}\ \mathit{ioA} \setminus \mathit{third} \\
\ \wedge \\
\ (\mathit{LocalC} \times \mathit{Value}) \setminus \operatorname{dom} \mathit{ioFinal\ ioC} \\
\qquad \subseteq \\
\qquad \operatorname{dom}((s \parallel \mathrm{id}) \rhd \operatorname{dom} \mathit{ioFinal\ ioA}\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet \qquad \text{[one point rule]} \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in S \mathbin{;} stopBatch^{T}\ ioA \setminus third \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC \\
\qquad \subseteq \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet \qquad \text{[def}^{\,n}\text{ of } \setminus\text{]} \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\,(c, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in S \mathbin{;} stopBatch^{T}\ ioA \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \notin third\,) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC \\
\qquad \subseteq \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet \qquad \text{[def}^{\,n}\text{ of } third\text{]} \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\,(c, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in S \mathbin{;} stopBatch^{T}\ ioA \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \mathrm{true}\,) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC \\
\qquad \subseteq \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet \qquad \text{[property of } \wedge\text{]} \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in S \mathbin{;} stopBatch^{T}\ ioA \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC \\
\qquad \subseteq \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,)
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{\,n}\ \text{of}\ \fatsemi] \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\,\exists\, a : LocalA;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad (c, \langle inF \rangle, outs) \mapsto (a, is, os) \in S & \\
\qquad\qquad \wedge & \\
\qquad\qquad (a, is, os) \mapsto outs \frown \langle outF \rangle \in stopBatch^{T}\ ioA\,) & \\
\quad \wedge & \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{\,n}\ \text{of} S] \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\,\exists\, a : LocalA;\ is, os : \mathrm{seq}\ Value \bullet & \\
\qquad\qquad c \mapsto a \in s \ \wedge\ is = \langle inF \rangle \ \wedge\ os = outs \frown \langle outF \rangle & \\
\qquad\qquad \wedge & \\
\qquad\qquad (a, is, os) \mapsto outs \frown \langle outF \rangle \in stopBatch^{T}\ ioA\,) & \\
\quad \wedge & \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet & [\text{one point rule}] \\
\qquad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\,\exists\, a : LocalA \bullet & \\
\qquad\qquad c \mapsto a \in s & \\
\qquad\qquad \wedge & \\
\qquad\qquad (a, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in stopBatch^{T}\ ioA\,) & \\
\quad \wedge & \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\qquad \subseteq & \\
\qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value;\ outs : \mathrm{seq}\ Value \bullet & [stopBatch^{T}] \\
\quad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a : LocalA \bullet & \\
\qquad\quad c \mapsto a \in s & \\
\qquad\quad \wedge & \\
\qquad\quad (a, \langle inF \rangle, outs) \mapsto outs \frown \langle outF \rangle \in stopBatch\ ioA\,) & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value \bullet & [stopBatch] \\
\quad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a : LocalA \bullet c \mapsto a \in s & \\
\qquad\quad \wedge & \\
\qquad\quad (a, inF) \mapsto outF \in ioFinal\ ioA\,) & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value \bullet & [\mathrm{def}^{\,n} \text{ of id}] \\
\quad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad \Rightarrow & \\
\qquad (\exists\, a : LocalA \bullet (c, inF) \mapsto (a, inF) \in (s \parallel \mathrm{id}) & \\
\qquad\quad \wedge & \\
\qquad\quad (a, inF) \mapsto outF \in ioFinal\ ioA\,) & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : LocalC;\ inF, outF : Value \bullet & [\mathrm{def}^{\,n} \text{ of } {}_{9}^{\circ}] \\
\quad (c, inF) \mapsto outF \in ioFinal\ ioC & \\
\qquad \Rightarrow & \\
\qquad (c, inF) \mapsto outF \in (s \parallel \mathrm{id}) \mathbin{{}_{9}^{\circ}} ioFinal\ ioA & \\
\wedge & \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioFinal\ ioC & \\
\quad \subseteq & \\
\quad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioFinal\ ioA\,) &
\end{array}
$$

$$\begin{array}{ll}\Leftrightarrow \mathit{ioFinal}\ \mathit{ioC} \subseteq (s \parallel \mathrm{id}) \fatsemi \mathit{ioFinal}\ \mathit{ioA} & [\mathrm{def}^n \text{ of } \subseteq] \\ \quad \wedge & \\ \quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\ \qquad \subseteq & \\ \qquad \mathrm{dom}((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}) & \end{array}$$

Lemma 6

$$\begin{array}{l}\forall n : \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC} \bullet \\ \quad \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T \ \subseteq\ S^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \\ \qquad \Leftrightarrow (\forall n : \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC} \bullet \\ \qquad\qquad \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi (s \parallel \mathrm{id}) \subseteq (s \parallel \mathrm{id}) \fatsemi \mathit{ioOps}\ \mathit{ioA}\ n \\ \qquad\qquad \wedge \\ \qquad\qquad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n \\ \qquad\qquad\quad \subseteq \\ \qquad\qquad\quad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n)) \end{array}$$

□

$$\mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T \subseteq S^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n$$

$$\begin{array}{ll}\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{property of } \subseteq] \\ \qquad\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T & \\ \qquad \subseteq & \\ \qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\ \quad \wedge & \\ \quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T & \\ \qquad \subseteq & \\ \qquad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd S^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \end{array}$$

$$\begin{array}{ll}\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } S^T] \\ \qquad\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T & \\ \qquad \subseteq & \\ \qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\ \quad \wedge & \\ \quad \mathit{BatchState}[\{\bot\}, \mathit{Value}] \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T & \\ \qquad \subseteq & \\ \qquad \mathrm{id}[\mathit{BatchState}[\{\bot\}, \mathit{Value}]] \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{def}^n \text{ of id}] \\
\qquad\qquad \lhd\, \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \,\comp\, S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \comp \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n & \\
\quad \wedge & \\
\quad \mathit{BatchState}[\{\perp\}, \mathit{Value}] \lhd \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \comp S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\{\perp\}, \mathit{Value}] \lhd \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{def}^n \text{ of } \subseteq] \\
\qquad\qquad \lhd\, \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \comp S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \comp \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \comp S^T & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{def}^n \text{ of } \Rightarrow] \\
\qquad\qquad \lhd\, \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \comp S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \comp \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\exists a : \mathit{LocalA}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad ((\perp, \mathit{ins}, \mathit{outs}) \mapsto (a, \mathit{is}, \mathit{os}) \in \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n & \\
\qquad\qquad\ \wedge & \\
\qquad\qquad\ (a, \mathit{is}, \mathit{os}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S^T)) & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{def}^n \text{ of } \mathit{batch}^T] \\
\qquad\qquad \lhd\, \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioC}\, n \comp S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \comp \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\exists a : \mathit{LocalA}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad (a = \perp\ \wedge\ \mathit{is} = \mathit{tail}\ \mathit{ins}\ \wedge\ \mathit{os} = \mathit{outs} & \\
\qquad\qquad\ \wedge & \\
\qquad\qquad\ (a, \mathit{is}, \mathit{os}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S^T)) & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\, \mathit{ioOps}\, \mathit{ioA}\, n &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{one point rule}] \\
\qquad \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (\perp, \mathit{tail}\ \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S^T & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } S^T] \\
\qquad \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad a' = \perp\ \wedge\ \mathit{tail}\ \mathit{ins} = \mathit{ins}'\ \wedge\ \mathit{outs} = \mathit{outs}' & \\
\qquad\quad \Rightarrow & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{property of } \Rightarrow] \\
\qquad \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad \forall a' : \mathit{LocalA}_\perp;\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad a' = \perp\ \wedge\ \mathit{tail}\ \mathit{ins} = \mathit{ins}'\ \wedge\ \mathit{outs} = \mathit{outs}' \bullet & \\
\qquad\quad (\perp, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{one point rule}] \\
\qquad \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad \forall \mathit{ins}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \mathit{tail}\ \mathit{ins} = \mathit{ins}' \bullet & \\
\qquad (\perp, \mathit{ins}, \mathit{outs}') \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \mathit{tail}] \\
\qquad \lhd \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T & \\
\qquad \subseteq & \\
\qquad \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd S^T \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad \forall \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \mathit{ins} = \langle \mathit{in} \rangle \frown \mathit{ins}' \bullet & \\
\qquad (\perp, \mathit{ins}, \mathit{outs}') \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad\qquad\qquad \text{[one point rule]} \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \\
& \quad \subseteq \\
& \quad BatchState[LocalC, Value] \lhd S^T \mathbin{;} batch^T\ ioOps\ ioA\ n \\
& \wedge \\
& \forall\, in : Value;\ ins, ins', outs' : \mathrm{seq}\ Value \bullet \\
& \quad (\bot, \langle in \rangle \frown ins', outs') \mapsto (\bot, ins', outs') \in batch^T\ ioOps\ ioA\ n
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad \text{[def}^n \text{ of } batch^T\ ioOps\ ioA\ n] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \\
& \quad \subseteq \\
& \quad BatchState[LocalC, Value] \lhd S^T \mathbin{;} batch^T\ ioOps\ ioA\ n \\
& \wedge \\
& \mathrm{true}
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad\qquad\qquad \text{[property of } \wedge] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \\
& \quad \subseteq \\
& \quad BatchState[LocalC, Value] \lhd S^T \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad\qquad\qquad \text{[def}^n \text{ of } S^T] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \\
& \quad \subseteq \\
& \quad S \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad\qquad\qquad \text{[property of } \subseteq] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
& \quad \subseteq \\
& \quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
& \wedge \\
& BatchState[LocalC, Value] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[\{\bot\}, Value] \\
& \quad \subseteq \\
& \quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[\{\bot\}, Value]
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow & BatchState[LocalC, Value] \qquad\qquad\qquad\qquad \text{[def}^n \text{ of } \subseteq] \\
& \quad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
& \quad \subseteq \\
& \quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
& \wedge \\
& \forall\, c : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet \\
& \quad (c, ins, outs) \mapsto (\bot, ins', outs') \in batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \\
& \qquad \Rightarrow \\
& \qquad (c, ins, outs) \mapsto (\bot, ins', outs') \in S \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \subseteq & \\
\quad S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\land & \\
\forall c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (\exists c' : \mathit{LocalC}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{is}, \mathit{os}) \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \land & \\
\qquad (c', \mathit{is}, \mathit{os}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in S^T) & \\
\qquad \Rightarrow & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } S^T] \\
\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \subseteq & \\
\quad S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\land & \\
\forall c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (\exists c' : \mathit{LocalC}_\perp;\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{is}, \mathit{os}) \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \land & \\
\qquad c' = \perp\ \land\ \mathit{is} = \mathit{ins}'\ \land\ \mathit{os} = \mathit{outs}') & \\
\qquad \Rightarrow & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{one point rule}] \\
\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \subseteq & \\
\quad S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\land & \\
\forall c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (c, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \Rightarrow & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \mathit{batch}^T] \\
\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \subseteq & \\
\quad S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\land & \\
\forall c : \mathit{LocalC};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\quad (c, \mathit{head}\ \mathit{ins}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n\ \land\ \mathit{ins}' = \mathit{tail}\ \mathit{ins}\ \land \mathit{outs}' = \mathit{outs} & \\
\qquad \Rightarrow & \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (\perp, \mathit{ins}', \mathit{outs}') \in S \mathbin{;} \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n &
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \qquad\qquad [\text{property of } \Rightarrow] \\
\qquad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
\quad \subseteq \\
\quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\wedge \\
\forall c : LocalC;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad ins' = tail\ ins\ \wedge outs' = outs \bullet \\
\quad (c, head\ ins) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \Rightarrow \\
\qquad (c, ins, outs) \mapsto (\bot, ins', outs') \in S \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \qquad\qquad [\mathrm{def}^n \text{ of } tail] \\
\qquad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
\quad \subseteq \\
\quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\wedge \\
\forall c : LocalC;\ in : Value;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad ins = \langle in \rangle \frown ins'\ \wedge outs' = outs \bullet \\
\quad (c, head\ ins) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \Rightarrow \\
\qquad (c, ins, outs) \mapsto (\bot, ins', outs') \in S \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \qquad\qquad [\text{one point rule}] \\
\qquad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
\quad \subseteq \\
\quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\wedge \\
\forall c : LocalC;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet \\
\quad (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \Rightarrow \\
\qquad (c, \langle in \rangle \frown ins', outs) \mapsto (\bot, ins', outs) \in S \mathbin{;} batch^T\ ioOps\ ioA\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \qquad\qquad [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad \lhd\, batch^T\ ioOps\ ioC\ n \mathbin{;} S^T \rhd BatchState[LocalA, Value] \\
\quad \subseteq \\
\quad S \mathbin{;} batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\wedge \\
\forall c : LocalC;\ in : Value;\ outs, ins' : \mathrm{seq}\ Value \bullet \\
\quad (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \Rightarrow \\
\qquad (\exists a : LocalA;\ is, os : \mathrm{seq}\ Value \bullet \\
\qquad\qquad (c, \langle in \rangle \frown ins', outs) \mapsto (a, is, os) \in S \\
\qquad\qquad \wedge \\
\qquad\qquad (a, is, os) \mapsto (\bot, ins', outs) \in batch^T\ ioOps\ ioA\ n\ )
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^{n} \text{ of } S] \\
\qquad\qquad \lhd\, \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^{T} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\ \wedge & \\
\ \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{outs}, \mathit{ins}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (\exists\, a : \mathit{LocalA};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad\qquad c \mapsto a \in s \ \wedge\ \mathit{is} = \langle \mathit{in} \rangle \frown \mathit{ins}' \ \wedge\ \mathit{os} = \mathit{outs} & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (a, \mathit{is}, \mathit{os}) \mapsto (\bot, \mathit{ins}', \mathit{outs}) \in \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioA}\ n\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\text{one point rule}] \\
\qquad\qquad \lhd\, \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^{T} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\ \wedge & \\
\ \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value};\ \mathit{outs}, \mathit{ins}' : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (\exists\, a : \mathit{LocalA} \bullet & \\
\qquad\qquad\qquad c \mapsto a \in s & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (a, \langle \mathit{in} \rangle \frown \mathit{ins}', \mathit{outs}) \mapsto (\bot, \mathit{ins}', \mathit{outs}) & \\
\qquad\qquad\qquad\qquad \in \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioA}\ n\,) &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^{n} \text{ of } \mathit{batch}^{T}] \\
\qquad\qquad \lhd\, \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} S^{T} \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\qquad \subseteq & \\
\qquad S \mathbin{;} \mathit{batch}^{T}\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\ \wedge & \\
\ \forall\, c : \mathit{LocalC};\ \mathit{in} : \mathit{Value} \bullet & \\
\qquad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (\exists\, a : \mathit{LocalA} \bullet c \mapsto a \in s & \\
\qquad\qquad\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad\qquad\qquad (a, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n\,) &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \hfill [\text{def}^n \text{ of id}] \\
\qquad\qquad \lhd\ batch^T\ ioOps\ ioC\ n \fatsemi S^T \rhd BatchState[LocalA, Value] \\
\qquad \subseteq \\
\qquad S \fatsemi batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value \bullet \\
\qquad (c, in) \notin \operatorname{dom} ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\exists a : LocalA \bullet (c, in) \mapsto (a, in) \in (s \parallel \mathrm{id}) \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (a, in) \notin \operatorname{dom} ioOps\ ioA\ n\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \hfill [\text{def}^n \text{ of } \ntriangleright] \\
\qquad\qquad \lhd\ batch^T\ ioOps\ ioC\ n \fatsemi S^T \rhd BatchState[LocalA, Value] \\
\qquad \subseteq \\
\qquad S \fatsemi batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value \bullet \\
\qquad (c, in) \notin \operatorname{dom} ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists a : LocalA \bullet \\
\qquad\qquad\qquad (c, in) \mapsto (a, in) \in (s \parallel \mathrm{id}) \ntriangleright \operatorname{dom} ioOps\ ioA\ n
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \hfill [\text{def}^n \text{ of dom}] \\
\qquad\qquad \lhd\ batch^T\ ioOps\ ioC\ n \fatsemi S^T \rhd BatchState[LocalA, Value] \\
\qquad \subseteq \\
\qquad S \fatsemi batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value \bullet \\
\qquad (c, in) \notin \operatorname{dom} ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, in) \in \operatorname{dom}((s \parallel \mathrm{id}) \ntriangleright \operatorname{dom} ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow BatchState[LocalC, Value] \hfill [\text{def}^n \text{ of } \setminus] \\
\qquad\qquad \lhd\ batch^T\ ioOps\ ioC\ n \fatsemi S^T \rhd BatchState[LocalA, Value] \\
\qquad \subseteq \\
\qquad S \fatsemi batch^T\ ioOps\ ioA\ n \rhd BatchState[LocalA, Value] \\
\quad \wedge \\
\quad \forall c : LocalC;\ in : Value \bullet \\
\qquad (c, in) \in (LocalC \times Value) \setminus \operatorname{dom} ioOps\ ioC\ n \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, in) \in \operatorname{dom}((s \parallel \mathrm{id}) \ntriangleright \operatorname{dom} ioOps\ ioA\ n)
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad\qquad \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S^T \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\qquad \subseteq & \\
\qquad S \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \mathbin{\rhd\!\!\!-} \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S & [\mathrm{def}^n \text{ of } S^T] \\
\qquad \subseteq & \\
\qquad S \fatsemi \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioA}\ n \rhd \mathit{BatchState}[\mathit{LocalA}, \mathit{Value}] & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \mathbin{\rhd\!\!\!-} \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \mathit{BatchState}[\mathit{LocalC}, \mathit{Value}] \lhd\ \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S & [\mathrm{def}^n \text{ of } \mathit{batch}^T] \\
\qquad \subseteq & \\
\qquad S \fatsemi \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \mathbin{\rhd\!\!\!-} \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & [\mathrm{def}^n \text{ of } \subseteq] \\
\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n \fatsemi S & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S \fatsemi \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \mathbin{\rhd\!\!\!-} \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \bullet & [\mathrm{def}^n \text{ of } \fatsemi] \\
\qquad \exists\, c' : \mathit{LocalC};\ \mathit{is}, \mathit{os} : \mathrm{seq}\ \mathit{Value} \bullet & \\
\qquad\qquad\quad (c, \mathit{ins}, \mathit{outs}) \mapsto (c', \mathit{is}, \mathit{os}) \in \mathit{batch}^T\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad (c', \mathit{is}, \mathit{os}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S \fatsemi \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \mathbin{\rhd\!\!\!-} \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet \qquad [\mathrm{def}^n \text{ of } S] \\
\qquad \exists\, c' : LocalC;\ is, os : \mathrm{seq}\ Value \bullet \\
\qquad\qquad\qquad (c, ins, outs) \mapsto (c', is, os) \in batch^T\ ioOps\ ioC\ n \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad c' \mapsto a' \in s \wedge is = ins' \wedge os = outs' \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet [\text{one point rule}] \\
\qquad \exists\, c' : LocalC \bullet (c, ins, outs) \mapsto (c', ins', outs') \in batch^T\ ioOps\ ioC\ n \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad c' \mapsto a' \in s \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \bullet [\mathrm{def}^n \text{ of } batch^T] \\
\qquad \exists\, c' : LocalC \bullet (c, ins, outs) \mapsto (c', ins', outs') \in batch\ ioOps\ ioC\ n \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad c' \mapsto a' \in s \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value\ \bullet \quad [\mathrm{def}^n \text{ of } batch] \\
\qquad \exists\, c' : LocalC \bullet (c, head\ ins) \mapsto (c', last\ outs') \in ioOps\ ioC\ n \\
\qquad\qquad \wedge \\
\qquad\qquad ins' = tail\ ins\ \wedge\ outs = first\ outs' \\
\qquad\qquad \wedge \\
\qquad\qquad c' \mapsto a' \in s \\
\qquad \Rightarrow \\
\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\wedge \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ ins, outs, ins', outs' : \mathrm{seq}\ Value \quad [\text{property of} \Rightarrow] \\
\qquad\qquad ins' = tail\ ins\ \wedge\ outs = first\ outs' \bullet \\
\qquad \exists\, c' : LocalC \bullet (c, head\ ins) \mapsto (c', last\ outs') \in ioOps\ ioC\ n \\
\qquad\qquad \wedge \\
\qquad\qquad c' \mapsto a' \in s \\
\qquad \Rightarrow \\
\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\wedge \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in : Value; \quad [\mathrm{def}^n \text{ of } tail] \\
\qquad\qquad\qquad ins, outs, ins', outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad ins = \langle in \rangle \frown ins'\ \wedge\ outs = first\ outs' \bullet \\
\qquad \exists\, c' : LocalC \bullet (c, head\ ins) \mapsto (c', last\ outs') \in ioOps\ ioC\ n \\
\qquad\qquad \wedge \\
\qquad\qquad c' \mapsto a' \in s \\
\qquad \Rightarrow \\
\qquad (c, ins, outs) \mapsto (a', ins', outs') \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\wedge \\
(LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{in}, \mathit{out} : \mathit{Value}; \qquad [\mathrm{def}^{\,n} \text{ of } \mathit{first}] \\
\qquad\qquad \mathit{ins}, \mathit{outs}, \mathit{ins}', \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad \mathit{ins} = \langle \mathit{in} \rangle \frown \mathit{ins}' \ \wedge\ \mathit{outs}' = \mathit{outs} \frown \langle \mathit{out} \rangle \bullet \\
\qquad \exists\, c' : \mathit{LocalC} \bullet (c, \mathit{head}\ \mathit{ins}) \mapsto (c', \mathit{last}\ \mathit{outs}') \in \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad\qquad \wedge \\
\qquad\qquad c' \mapsto a' \in s \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \mathit{ins}, \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs}') \in S \mathbin{;} \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n \\
\wedge \\
(\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{in}, \mathit{out} : \mathit{Value}; \qquad [\text{one point rule}] \\
\qquad\qquad \mathit{outs}, \mathit{ins}' : \mathrm{seq}\ \mathit{Value} \bullet \\
\qquad \exists\, c' : \mathit{LocalC} \bullet (c, \mathit{in}) \mapsto (c', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad\qquad \wedge \\
\qquad\qquad c' \mapsto a' \in s \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \langle \mathit{in} \rangle \frown \mathit{ins}', \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs} \frown \langle \mathit{out} \rangle) \\
\qquad\qquad \in S \mathbin{;} \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n \\
\wedge \\
(\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{in}, \mathit{out} : \mathit{Value}; \qquad [\text{property of id}] \\
\qquad\qquad \mathit{outs}, \mathit{ins}' : \mathrm{seq}\ \mathit{Value} \bullet \\
\qquad \exists\, c' : \mathit{LocalC} \bullet (c, \mathit{in}) \mapsto (c', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad\qquad \wedge \\
\qquad\qquad (c', \mathit{out}) \mapsto (a', \mathit{out}) \in (s \parallel \mathrm{id}) \\
\qquad\quad \Rightarrow \\
\qquad\quad (c, \langle \mathit{in} \rangle \frown \mathit{ins}', \mathit{outs}) \mapsto (a', \mathit{ins}', \mathit{outs} \frown \langle \mathit{out} \rangle) \\
\qquad\qquad \in S \mathbin{;} \mathit{batch}\ \mathit{ioOps}\ \mathit{ioA}\ n \\
\wedge \\
(\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value; \qquad\qquad [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (c, \langle in \rangle \frown ins', outs) \mapsto (a', ins', outs \frown \langle out \rangle) \\
\qquad\qquad\qquad \in S \mathbin{;} batch\ ioOps\ ioA\ n \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value; \qquad\qquad [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\ \exists\, a : LocalA;\ is, os : \mathrm{seq}\ Value \bullet \\
\qquad\qquad\qquad (c, \langle in \rangle \frown ins', outs) \mapsto (a, is, os) \in S \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, is, os) \mapsto (a', ins', outs \frown \langle out \rangle) \in batch\ ioOps\ ioA\ n\ ) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value; \qquad\qquad [\mathrm{def}^n \text{ of } S] \\
\qquad\qquad\qquad\qquad\qquad outs, ins' : \mathrm{seq}\ Value \bullet \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\ \exists\, a : LocalA;\ is, os : \mathrm{seq}\ Value \bullet \\
\qquad\qquad\qquad c \mapsto a \in s\ \wedge\ is = \langle in \rangle \frown ins'\ \wedge\ os = outs \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, is, os) \mapsto (a', ins', outs \frown \langle out \rangle) \in batch\ ioOps\ ioA\ n\ ) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \mathrm{dom}\ ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ ioOps\ ioA\ n)
\end{array}
$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value; \qquad\qquad [\text{one point rule}] \\
\qquad\qquad\qquad\qquad\qquad outs, ins' : \operatorname{seq} Value \bullet \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\,\exists\, a : LocalA \bullet \\
\qquad\qquad\qquad c \mapsto a \in s \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, \langle in \rangle \frown ins', outs) \mapsto (a', ins', outs \frown \langle out \rangle) \\
\qquad\qquad\qquad\quad \in \\
\qquad\qquad\qquad\quad batch\ ioOps\ ioA\ n\,) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioOps\ ioA\ n)
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value \bullet \qquad\qquad [\mathrm{def}^{n} \text{ of } batch] \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\,\exists\, a : LocalA \bullet \\
\qquad\qquad\qquad c \mapsto a \in s \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, in) \mapsto (a', out) \in ioOps\ ioA\ n\,) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioOps\ ioA\ n)
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, c : LocalC;\ a' : LocalA;\ in, out : Value \bullet \qquad\qquad [\text{property of id}] \\
\qquad (c, in) \mapsto (a', out) \in ioOps\ ioC\ n \mathbin{;} (s \parallel \mathrm{id}) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\,\exists\, a : LocalA \bullet \\
\qquad\qquad\qquad (c, in) \mapsto (a, in) \in (s \parallel \mathrm{id}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, in) \mapsto (a', out) \in ioOps\ ioA\ n\,) \\
\quad \wedge \\
\quad (LocalC \times Value) \setminus \operatorname{dom} ioOps\ ioC\ n \\
\qquad \subseteq \\
\qquad ((s \parallel \mathrm{id}) \rhd \operatorname{dom} ioOps\ ioA\ n)
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, c : \mathit{LocalC};\ a' : \mathit{LocalA};\ \mathit{in}, \mathit{out} : \mathit{Value} \bullet & [\mathrm{def}^{\,n} \text{ of } \mathbin{;}] \\
\qquad (c, \mathit{in}) \mapsto (a', \mathit{out}) \in \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} (s \parallel \mathrm{id}) & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad (c, \mathit{in}) \mapsto (a', \mathit{out}) \in (s \parallel \mathrm{id}) \mathbin{;} \mathit{ioOps}\ \mathit{ioA}\ n & \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) & \\
\\
\Leftrightarrow \mathit{ioOps}\ \mathit{ioC}\ n \mathbin{;} (s \parallel \mathrm{id}) \subseteq (s \parallel \mathrm{id}) \mathbin{;} \mathit{ioOps}\ \mathit{ioA}\ n & [\mathrm{def}^{\,n} \text{ of } \subseteq] \\
\quad \wedge & \\
\quad (\mathit{LocalC} \times \mathit{Value}) \setminus \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ n & \\
\qquad \subseteq & \\
\qquad ((s \parallel \mathrm{id}) \rhd \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ n) &
\end{array}
$$

Appendix B

Simulation of ioData types implies refinement

In this section we prove the soundness of the forwards and backwards simulation rules for ioData types.

B.1 Forwards simulation rules

Lemma 7 Given two ioData types, if the first forwards simulates the second then the first is refined by the second. □

Proof

Let ioA and ioC be ioData types of type $ioData[LocalA, Name, Value]$ and $ioData[LocalC, Name, Value]$ respectively such that ioA forwards simulates ioC that is, there exists a retrieve relation R such that the following three Conditions hold.

1. $startBatch^T\ ioC \ \subseteq\ startBatch^T\ ioA \mathbin{;} R^T$
2. $R^T \mathbin{;} stopBatch^T\ ioC \ \subseteq\ stopBatch^T\ ioA$
3. $\forall\, n : \mathrm{dom}\, ioOps\ ioC \bullet$
   $R^T \mathbin{;} batch^T\ ioOps\ ioC\ n \ \subseteq\ batch^T\ ioOps\ ioA\ n \mathbin{;} R^T$

First we will prove by induction on the length of the sequence that for all sequences $p : \mathrm{seq}(\mathrm{dom}\, ioOps\ ioC)$ the following is true.

$$R^T \mathbin{⨾} runBatch^T\ ioC\ p \subseteq runBatch^T\ ioA\ p \mathbin{⨾} R^T$$

Given the truth set $\mathcal{T}$, our inductive hypothesis, is as follows:

$$\begin{array}{l} \forall\, n : \mathbb{N} \bullet \\ \quad n \in \mathcal{T} \Leftrightarrow (\forall\, p : \mathrm{seq}(\mathrm{dom}\, ioOps\ ioC) \mid \#p \leq n \bullet \\ \qquad\qquad R^T \mathbin{⨾} runBatch^T\ ioC\ p \subseteq runBatch^T\ ioA\ p \mathbin{⨾} R^T\,) \end{array}$$

The base case, when p is the empty sequence is trivially true since

$$\begin{array}{lcl} runBatch^T\ ioC\ \ \langle\rangle & = & \mathrm{id}[LocalC_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value] \\ \text{and} & & \\ runBatch^T\ ioA\ \ \langle\rangle & = & \mathrm{id}[LocalA_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value] \end{array}$$

and hence

$$R^T \mathbin{⨾} runBatch^T\ ioC\ \ \langle\rangle = R^T \quad \text{and} \quad runBatch^T\ ioA\ \ \langle\rangle \mathbin{⨾} R^T = R^T.$$

and so we have shown that $0 \in \mathcal{T}$.

Now, suppose that our inductive hypothesis holds for some natural number k and that p_k is a sequence of type $\mathrm{seq}(\mathrm{dom}\, ioOps\ ioC)$ of length k, that op is the name of an operation on ioC, that a and c' are elements of types $LocalA_\perp$ and $LocalC_\perp$ respectively and that ins,$outs$,ins' and $outs'$ are sequences of type $\mathrm{seq}\ Value$.

$$\begin{array}{ll} (a, ins, outs) \mapsto (c', ins', outs') \in R^T \mathbin{⨾} runBatch^T\ ioC\ (p_k \frown \langle op\rangle) & \\ \quad \Leftrightarrow (a, ins, outs) \mapsto (c', ins', outs') & [\mathrm{def}^n \text{ of } runBatch^T] \\ \qquad\quad \in R^T \mathbin{⨾} runBatch^T\ ioC\ p_k \mathbin{⨾} batch^T\ (ioOps\ ioC\ op) & \\ \quad \Leftrightarrow \exists\, c : LocalC_\perp;\ is, os : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } ⨾] \\ \qquad\quad (a, ins, outs) \mapsto (c, is, os) \in R^T \mathbin{⨾} runBatch^T\ ioC\ p_k & \\ \qquad\quad \wedge & \\ \qquad\quad (c, is, os) \mapsto (c', ins', outs') \in batch^T\ (ioOps\ ioC\ op) & \\ \quad \Rightarrow \exists\, c : LocalC_\perp;\ is, os : \mathrm{seq}\ Value \bullet & [\text{inductive hypothesis}] \\ \qquad\quad (a, ins, outs) \mapsto (c, is, os) \in runBatch^T\ ioA\ p_k \mathbin{⨾} R^T & \\ \qquad\quad \wedge & \\ \qquad\quad (c, is, os) \mapsto (c', ins', outs') \in batch^T\ (ioOps\ ioC\ op) & \end{array}$$

$$\Leftrightarrow (a, ins, outs) \mapsto (c', ins', outs') \qquad [\text{def}^n \text{ of } \fatsemi]$$
$$\in runBatch^T\ ioA\ p_k \fatsemi R^T \fatsemi batch^T\ (ioOps\ ioC\ op)$$

$$\Leftrightarrow \exists\, a' : LocalA_\perp;\ is', os' : \text{seq}\ Value \bullet \qquad [\text{def}^n \text{ of } \fatsemi]$$
$$(a, ins, outs) \mapsto (a', is', os') \in runBatch^T\ ioA\ p_k$$
$$\wedge$$
$$(a', is', os') \mapsto (c', ins', outs') \in R^T \fatsemi batch^T\ (ioOps\ ioC\ op)$$

$$\Rightarrow \exists\, a' : LocalA_\perp;\ is', os' : \text{seq}\ Value \bullet \qquad [\text{Condition 3}]$$
$$(a, ins, outs) \mapsto (a', is', os') \in runBatch^T\ ioA\ p_k$$
$$\wedge$$
$$(a', is', os') \mapsto (c', ins', outs') \in batch^T\ (ioOps\ ioC\ op) \fatsemi R^T$$

$$\Leftrightarrow (a, ins, outs) \mapsto (c', ins', outs') \qquad [\text{def}^n \text{ of } \fatsemi]$$
$$\in runBatch^T\ ioA\ p_k \fatsemi batch^T\ (ioOps\ ioC\ op) \fatsemi R^T$$

$$\Leftrightarrow (a, ins, outs) \mapsto (c', ins', outs') \qquad [\text{def}^n \text{ of } runBatch^T]$$
$$\in runBatch^T\ ioA\ (p_k \frown \langle op \rangle) \fatsemi R^T$$

This holds for all operations op and hence we have shown that

$$k \in \mathcal{T} \ \Rightarrow\ (k+1) \in \mathcal{T}$$

Hence, we have proved that if ioA forwards simulates ioC then Condition 4 below must be true.

$$\forall\, p : \text{seq}(domioOps\ ioC) \bullet$$
$$R^T \fatsemi runBatch^T\ ioC\ p \ \subseteq\ runBatch^T\ ioA\ p \fatsemi R^T$$

We now use this result to show that if ioA forwards simulates ioC than ioA is refined by ioC.

Let p be a sequence of type seq(dom $ioOps\ ioC$), inI and $outI$ be of type $Value$ and ins and $outs'$ be sequences of type seq $Value$ such that ins is the same length as p.

$$\langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs'$$
$$\in startBatch^T\ ioC \fatsemi runBatch^T\ ioC\ p \fatsemi stopBatch^T\ ioC$$

$$\Leftrightarrow \exists\, c : LocalC_\perp;\ os : \text{seq}\ Value\ \bullet \qquad [\text{def}^n \text{ of } \fatsemi]$$
$$\langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, ins \frown \langle inF \rangle, os)$$
$$\in startBatch^T\ ioC$$
$$\wedge$$
$$(c, ins \frown \langle inF \rangle, os) \mapsto outs'$$
$$\in runBatch^T\ ioC\ p \fatsemi stopBatch^T\ ioC$$

$$
\begin{array}{ll}
\Rightarrow \exists\, c : LocalC_{\perp};\ os : \mathrm{seq}\ Value \bullet & [\text{Condition 1}] \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, ins \frown \langle inF \rangle, os) & \\
\quad\quad \in startBatch^{T}\ ioA \mathbin{;} R^{T} & \\
\quad \wedge & \\
\quad (c, ins \frown \langle inF \rangle, os) \mapsto outs' & \\
\quad\quad \in runBatch^{T}\ ioC\ p \mathbin{;} stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Rightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\mathrm{def}^{n} \text{ of } ;] \\
\quad \in (startBatch^{T}\ ioA \mathbin{;} R^{T}) \mathbin{;} runBatch^{T}\ ioC\ p \mathbin{;} stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\text{associativity of } ;] \\
\quad \in startBatch^{T}\ ioA \mathbin{;} (R^{T} \mathbin{;} runBatch^{T}\ ioC\ p) \mathbin{;} stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \exists\, a : LocalA_{\perp};\ c' : LocalC_{\perp};\ os, os' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n} \text{ of } ;] \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, ins \frown \langle inF \rangle, os) & \\
\quad\quad \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad (a, ins \frown \langle inF \rangle, os) \mapsto (c', \langle inF \rangle, os') & \\
\quad\quad \in R^{T} \mathbin{;} runBatch^{T}\ ioC\ p & \\
\quad \wedge & \\
\quad (c', \langle inF \rangle, os') \mapsto outs' & \\
\quad\quad \in stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Rightarrow \exists\, a : LocalA_{\perp};\ c' : LocalC_{\perp};\ os, os' : \mathrm{seq}\ Value \bullet & [\text{Condition 4}] \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, ins \frown \langle inF \rangle, os) & \\
\quad\quad \in startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad (a, ins \frown \langle inF \rangle, os) \mapsto (c', \langle inF \rangle, os') & \\
\quad\quad \in runBatch^{T}\ ioA\ p \mathbin{;} R^{T} & \\
\quad \wedge & \\
\quad (c', \langle inF \rangle, os') \mapsto outs' & \\
\quad\quad \in stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\mathrm{def}^{n} \text{ of } ;] \\
\quad \in startBatch^{T}\ ioA \mathbin{;} (runBatch^{T}\ ioA\ p \mathbin{;} R^{T}) \mathbin{;} stopBatch^{T}\ ioC &
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\text{associativity of } ;] \\
\quad \in startBatch^{T}\ ioA \mathbin{;} runBatch^{T}\ ioA\ p \mathbin{;} (R^{T} \mathbin{;} stopBatch^{T}\ ioC) &
\end{array}
$$

$$\Leftrightarrow \exists\, a' : \mathit{LocalA}_\perp;\ os : \mathrm{seq}\ \mathit{Value} \bullet \qquad [\mathrm{def}^n \text{ of } \mathbin{;}]$$
$$\langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a', \langle \mathit{inF} \rangle, os)$$
$$\in \mathit{startBatch}^T\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^T\ \mathit{ioA}\ p$$
$$\wedge$$
$$(a', \langle \mathit{inF} \rangle, os) \mapsto \mathit{outs}'$$
$$\in R^T \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}$$

$$\Rightarrow \exists\, a' : \mathit{LocalA}_\perp;\ os : \mathrm{seq}\ \mathit{Value} \bullet \qquad [\text{Condition } 2]$$
$$\langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a', \langle \mathit{inF} \rangle, os)$$
$$\in \mathit{startBatch}^T\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^T\ \mathit{ioA}\ p$$
$$\wedge$$
$$(a', \langle \mathit{inF} \rangle, os) \mapsto \mathit{outs}'$$
$$\in \mathit{stopBatch}^T\ \mathit{ioA}$$

$$\Rightarrow \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto \mathit{outs}' \qquad [\mathrm{def}^n \text{ of } \mathbin{;}]$$
$$\in \mathit{startBatch}^T\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^T\ \mathit{ioA}\ p \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA}$$

This holds for all p, inI, outI, ins, outs' described above and so we conclude

$$\forall\, p : \mathrm{seq}(\mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}) \bullet$$
$$\mathit{startBatch}^T\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^T\ \mathit{ioC}\ p \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioC}$$
$$\subseteq$$
$$\mathit{startBatch}^T\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^T\ \mathit{ioA}\ p \mathbin{;} \mathit{stopBatch}^T\ \mathit{ioA}$$

or equivalently

$$\mathit{ioSemantics}\ \mathit{ioC} \ \subseteq\ \mathit{ioSemantics}\ \mathit{ioA}$$

or equivalently, ioA is refined by ioC.

B.2 Backwards simulation rules

Lemma 8 Given two ioData types, if the first backwards simulates the second then the first is refined by the second. □

Proof

Let ioA and ioC be ioData types of type $\mathit{ioData}[\mathit{LocalA}, \mathit{Name}, \mathit{Value}]$ and $\mathit{ioData}[\mathit{LocalC}, \mathit{Name}, \mathit{Value}]$ respectively such that ioA backwards simulates ioC that is, there exists a retrieve relation S such that the following three Conditions hold.

1. $startBatch^T\ ioC \fatsemi S^T \subseteq startBatch^T\ ioA$
2. $stopBatch^T\ ioC \subseteq S^T \fatsemi stopBatch^T\ ioA$
3. $\forall n : \mathrm{dom}\ ioOps\ ioC \bullet$
   $batch^T\ ioOps\ ioC\ n \fatsemi S^T \subseteq S^T \fatsemi batch^T\ ioOps\ ioA\ n$

First we will prove by induction on the length of the sequence that for all sequences $p : \mathrm{seq}(\mathrm{dom}\ ioOps\ ioC)$ the following is true.

$$runBatch^T\ ioC\ p \fatsemi S^T \subseteq S^T \fatsemi runBatch^T\ ioA\ p$$

Given the truth set $\mathcal{T}$, our inductive hypothesis, is as follows:

$$\begin{array}{l} \forall n : \mathbb{N} \bullet \\ \quad n \in \mathcal{T} \Leftrightarrow (\forall p : \mathrm{seq}(\mathrm{dom}\ ioOps\ ioC) \mid \#p \leq n \bullet \\ \qquad\qquad\qquad runBatch^T\ ioC\ p \fatsemi S^T \subseteq S^T \fatsemi runBatch^T\ ioA\ p\ ) \end{array}$$

The base case, when p is the empty sequence is trivially true since

$$\begin{array}{l} runBatch^T\ ioC\ \langle\rangle = \mathrm{id}[LocalC_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value] \\ \text{and } runBatch^T\ ioA\ \langle\rangle = \mathrm{id}[LocalA_\perp \times \mathrm{seq}\ Value \times \mathrm{seq}\ Value] \end{array}$$

and hence

$$runBatch^T\ ioC\ \langle\rangle \fatsemi S^T = S^T \text{ and } S^T \fatsemi runBatch^T\ ioA\ \langle\rangle = S^T.$$

and so we have shown that $0 \in \mathcal{T}$.

Now, suppose that our inductive hypothesis holds for some natural number k and that p_k is a sequence of type $\mathrm{seq}(\mathrm{dom}\ ioOps\ ioC)$ of length k, that op is the name of an operation on ioC, that a' and c are elements of types $LocalA_\perp$ and $LocalC_\perp$ respectively and that ins, $outs$, ins' and $outs'$ are sequences of type $\mathrm{seq}\ Value$.

$$(c, ins, outs) \mapsto (a', ins', outs') \in runBatch^T\ ioC\ (p_k \frown \langle op\rangle) \fatsemi S^T$$

$$\begin{array}{ll} \Leftrightarrow (c, ins, outs) \mapsto (a', ins', outs') & [\mathrm{def}^n \text{ of } runBatch^T] \\ \qquad \in runBatch^T\ ioC\ p_k \fatsemi batch^T\ (ioOps\ ioC\ op) \fatsemi S^T & \end{array}$$

$$\begin{array}{ll} \Leftrightarrow \exists c' : LocalC_\perp;\ is', os' : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } \fatsemi] \\ \qquad (c, ins, outs) \mapsto (c', is', os') & \\ \qquad\quad \in runBatch^T\ ioC\ p_k & \\ \qquad \wedge & \\ \qquad (c', is', os') \mapsto (a', ins', outs') & \\ \qquad\quad \in batch^T\ (ioOps\ ioC\ op) \fatsemi S^T & \end{array}$$

$$
\begin{array}{ll}
\Rightarrow \exists\, c' : LocalC_\perp;\ is', os' : \mathrm{seq}\ Value \bullet & \text{[Condition 4]} \\
\qquad (c, ins, outs) \mapsto (c', is', os') & \\
\qquad\qquad \in runBatch^T\ ioC\ p_k & \\
\qquad \wedge & \\
\qquad (c', is', os') \mapsto (a', ins', outs') & \\
\qquad\qquad \in S^T \mathbin{;} batch^T\ (ioOps\ ioA\ op) & \\[1ex]
\Leftrightarrow (c, ins, outs) \mapsto (a', ins', outs') & [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad \in runBatch^T\ ioC\ p_k \mathbin{;} S^T \mathbin{;} batch^T\ (ioOps\ ioA\ op) & \\[1ex]
\Leftrightarrow \exists\, a : LocalA_\perp;\ is, os : \mathrm{seq}\ Value \bullet & [\mathrm{def}^n \text{ of } \mathbin{;}] \\
\qquad (c, ins, outs) \mapsto (a, is, os) & \\
\qquad\qquad \in runBatch^T\ ioC\ p_k \mathbin{;} S^T & \\
\qquad \wedge & \\
\qquad (a, is, os) \mapsto (a', ins', outs') & \\
\qquad\qquad \in batch^T\ (ioOps\ ioA\ op) & \\[1ex]
\Rightarrow \exists\, a : LocalA_\perp;\ is, os : \mathrm{seq}\ Value \bullet & \text{[inductive hypothesis]} \\
\qquad (c, ins, outs) \mapsto (a, is, os) & \\
\qquad\qquad \in S^T \mathbin{;} runBatch^T\ ioA\ p_k & \\
\qquad \wedge & \\
\qquad (a, is, os) \mapsto (a', ins', outs') & \\
\qquad\qquad \in batch^T\ (ioOps\ ioA\ op) & \\[1ex]
\Leftrightarrow (c, ins, outs) \mapsto (a', ins', outs') & [\mathrm{def}^n \text{ of } \exists] \\
\qquad \in S^T \mathbin{;} runBatch^T\ ioA\ p_k \mathbin{;} batch^T\ (ioOps\ ioA\ op) & \\[1ex]
\Leftrightarrow (c, ins, outs) \mapsto (a', ins', outs') & [\mathrm{def}^n \text{ of } runBatch^T] \\
\qquad \in S^T \mathbin{;} runBatch^T\ ioA\ (p_k \frown \langle op \rangle) &
\end{array}
$$

This holds for all operations op and hence we have shown that

$$k \in \mathcal{T} \ \Rightarrow\ (k+1) \in \mathcal{T}$$

Hence, we have proved that if ioA backwards simulates ioC then Condition 4 below must be true.

$$
\begin{array}{l}
\forall\, p : \mathrm{seq}(domioOps\ ioC) \bullet \\
\qquad runBatch^T\ ioC\ p \mathbin{;} S^T \ \subseteq\ S^T \mathbin{;} runBatch^T\ ioA\ p
\end{array}
$$

We now use this result to show that if ioA backwards simulates ioC than ioA is refined by ioC.

Let p be a sequence of type $\operatorname{seq}(\operatorname{dom} ioOps\ ioC)$, inI and $outI$ be of type $Value$ and ins and $outs'$ be sequences of type $\operatorname{seq} Value$ such that ins is the same length as p.

$$\begin{array}{l}
\langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' \\
\quad \in startBatch^T\ ioC \fatsemi runBatch^T\ ioC\ p \fatsemi stopBatch^T\ ioC \\[1em]
\quad \Leftrightarrow \exists\, c' : LocalC_\perp;\ is', os' : \operatorname{seq} Value \bullet \qquad [\text{def}^n \text{ of } \fatsemi] \\
\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c', is', os') \\
\qquad\qquad \in startBatch^T\ ioC \fatsemi runBatch^T\ ioC\ p \\
\qquad\quad \wedge \\
\qquad\quad (c', is', os') \mapsto outs' \\
\qquad\qquad \in stopBatch^T\ ioC \\[1em]
\quad \Rightarrow \exists\, c' : LocalC_\perp;\ is', os' : \operatorname{seq} Value \bullet \qquad [\text{Condition 3}] \\
\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c', is', os') \\
\qquad\qquad \in startBatch^T\ ioC \fatsemi runBatch^T\ ioC\ p \\
\qquad\quad \wedge \\
\qquad\quad (c', is', os') \mapsto outs' \\
\qquad\qquad \in S^T \fatsemi stopBatch^T\ ioA \\[1em]
\quad \Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' \qquad [\text{def}^n \text{ of } \fatsemi] \\
\qquad\quad \in startBatch^T\ ioC \fatsemi runBatch^T\ ioC\ p \fatsemi (S^T \fatsemi stopBatch^T\ ioA) \\[1em]
\quad \Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' \qquad [\text{associativity}] \\
\qquad\quad \in startBatch^T\ ioC \fatsemi (runBatch^T\ ioC\ p \fatsemi S^T) \fatsemi stopBatch^T\ ioA \\[1em]
\quad \Leftrightarrow \exists\, c : LocalC_\perp;\ a' : LocalA_\perp;\ is, os, is', os' : \operatorname{seq} Value \bullet \qquad [\text{def}^n \text{ of } \fatsemi] \\
\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, is, os) \\
\qquad\qquad \in startBatch^T\ ioC \\
\qquad\quad \wedge \\
\qquad\quad (c, is, os) \mapsto (a', is', os') \\
\qquad\qquad \in runBatch^T\ ioC\ p \fatsemi S^T \\
\qquad\quad \wedge \\
\qquad\quad (a', is', os') \mapsto outs' \\
\qquad\qquad \in stopBatch^T\ ioA
\end{array}$$

$$\begin{array}{ll}
\Rightarrow \exists\, c : LocalC_{\perp};\ a' : LocalA_{\perp};\ is, os, is', os' : \mathrm{seq}\ Value \bullet & \text{[Condition 4]} \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, is, os) & \\
\qquad \in startBatch^{T}\ ioC & \\
\quad \wedge & \\
\quad (c, is, os) \mapsto (a', is', os') & \\
\qquad \in S^{T} \fatsemi runBatch^{T}\ ioA\ p & \\
\quad \wedge & \\
\quad (a', is', os') \mapsto outs' & \\
\qquad \in stopBatch^{T}\ ioA & \\
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\mathrm{def}^{n}\ \mathrm{of}\ \fatsemi] \\
\quad \in startBatch^{T}\ ioC \fatsemi (S^{T} \fatsemi runBatch^{T}\ ioA\ p) \fatsemi stopBatch^{T}\ ioA & \\
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & \text{[associativity]} \\
\quad \in (startBatch^{T}\ ioC \fatsemi S^{T}) \fatsemi runBatch^{T}\ ioA\ p \fatsemi stopBatch^{T}\ ioA & \\
\Leftrightarrow \exists\, a : LocalA_{\perp};\ is, os : \mathrm{seq}\ Value \bullet & [\mathrm{def}^{n}\ \mathrm{of}\ \fatsemi] \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, is, os) & \\
\qquad \in startBatch^{T}\ ioC \fatsemi S^{T} & \\
\quad \wedge & \\
\quad (a, is, os) \mapsto outs' & \\
\qquad \in runBatch^{T}\ ioA\ p \fatsemi stopBatch^{T}\ ioA & \\
\Rightarrow \exists\, a : LocalA_{\perp};\ is, os : \mathrm{seq}\ Value \bullet & \text{[Condition 1]} \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, is, os) & \\
\qquad \in S^{T} \fatsemi startBatch^{T}\ ioA & \\
\quad \wedge & \\
\quad (a, is, os) \mapsto outs' & \\
\qquad \in runBatch^{T}\ ioA\ p \fatsemi stopBatch^{T}\ ioA & \\
\Leftrightarrow \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto outs' & [\mathrm{def}^{n}\ \mathrm{of}\ \fatsemi] \\
\quad \in S^{T} \fatsemi startBatch^{T}\ ioA \fatsemi runBatch^{T}\ ioA\ p \fatsemi stopBatch^{T}\ ioA &
\end{array}$$

This holds for all p, inI, $outI$, ins, $outs'$ described above and so we conclude

$$\begin{array}{l}
\forall\, p : \mathrm{seq}(\mathrm{dom}\ ioOps\ ioC) \bullet \\
\quad startBatch^{T}\ ioC \fatsemi runBatch^{T}\ ioC\ p \fatsemi stopBatch^{T}\ ioC \\
\qquad \subseteq \\
\qquad startBatch^{T}\ ioA \fatsemi runBatch^{T}\ ioA\ p \fatsemi stopBatch^{T}\ ioA
\end{array}$$

or equivalently

$$ioSemantics\ ioC \ \subseteq\ ioSemantics\ ioA$$

or equivalently, ioA is refined by ioC.

We have shown that if one ioData type backwards simulates another then the first is refined by the second.

Appendix C

Refinement of ioData types and their process equivalents

In this chapter we prove Theorem 5 which stated that one ioData type is refined by another precisely when the ioProcess corresponding to the first is refined by the ioProcess corresponding to the second.

First we establish three results working from the failures refinement of ioProcesses end of the proof; the first considering failures after the empty trace, the second considering failures after a finished trace and the third, and most complex, considering failures after an unfinished trace.

We then call upon these three results in order to establish the equivalence working from the the ioData type end of the proof.

We consider only ioData types with a single operation op, that is

$$\mathrm{dom}\, \mathit{ioOps}\, \mathit{ioA} \ = \ \mathrm{dom}\ \mathit{ioOps}\, \mathit{ioC} \ = \ \{op\}$$

The result may be generalised to extend to all ioData types and their corresponding ioProcesses.

C.1 Failures refinement

Recall that one process is failures refined by another precisely when the failures of the second is a subset of the failures of the first.

Furthermore, recall that the failures of an ioProcess may be expressed as the disjoint union of the failures of the ioProcess after the empty trace, the failures of the ioProcess after all finished traces and the failures of the ioProcess after all unfinished traces.

It follows from these two observations that one ioProcess is failures refined by another precisely when the following conditions hold:

- the failures of the second ioProcess after the empty trace is a subset of the failures of the first ioProcess after the empty trace.
- the failures of the second ioProcess after all finished traces is a subset of the failures of the first ioProcess after all finished traces.
- the failures of the second ioProcess after all unfinished traces is a subset of the failures of the first ioProcess after all unfinished traces.

In this section we present three lemmas, each simplifying the predicate associated with one of the above conditions.

Failures after the empty trace

First we consider failures after the empty trace.

We recall that the failures for $\mathsf{ioProcess}\,(ioA)$ (corresponding to ioData type ioA) after the empty trace is given by

$$
\begin{array}{l}
\{X : \mathbb{P}\,\Sigma \mid \\
\quad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\ ioA \bullet \\
\quad\quad \mathrm{init}.inI.outI \in X \\
\quad\quad\quad \Rightarrow \\
\quad\quad\quad \exists\, outI' : \operatorname{ran}(ioInit\ ioA \,(\!|\, \{inI\} \,|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\quad\quad\quad\quad \bullet\, (\,\langle\rangle\,, X)\}
\end{array}
$$

We now present a lemma which asserts that the failures of $\mathsf{ioProcess}\,(ioC)$ after the empty trace is always a subset of the failures of $\mathsf{ioProcess}\,(ioA)$ after the empty trace

Lemma 9
Given ioData types ioA and ioC of type $ioData[Value, Name, Local]$ such that

- $\operatorname{dom} ioInit\ ioA = \operatorname{dom} ioInit\ ioC$

- $\forall \, inI \; : \; \mathrm{dom}\, ioInit\; ioC \; \bullet \; \mathrm{ran}\, ioInit\; ioC (\!|\; \{inI\} \;|\!) \subseteq \mathrm{ran}\, ioInit\; ioA (\!|\; \{inI\} \;|\!)$

the failures set of ioProcess (ioC) at initialisation is always a subset of the failures set at initialisation of ioProcess (ioA), that is

$$
\begin{array}{l}
(\{X : \mathbb{P}\,\Sigma \mid \\
\quad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\; ioC \bullet \\
\quad\quad \mathrm{init}.inI.outI \in X \\
\quad\quad\quad \Rightarrow \\
\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\; ioC (\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\quad\quad\quad\quad \bullet (\langle\rangle, X)\} \\
\quad \subseteq \\
\quad \{X : \mathbb{P}\,\Sigma \mid \\
\quad\quad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\; ioA \bullet \\
\quad\quad\quad \mathrm{init}.inI.outI \in X \\
\quad\quad\quad\quad \Rightarrow \\
\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\; ioA (\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\quad\quad\quad\quad\quad \bullet (\langle\rangle, X)\}) \\
\Leftrightarrow \\
\mathrm{true}
\end{array}
$$

$\Box$

Proof

First we show that given a set $Y : \mathbb{P}\,\Sigma$, if Y is a refusals set of ioProcess (ioC) at initialisation then Y must be a refusals set of ioProcess (ioA) at initialisation.

We then use this to establish the desired result.

$$
\begin{array}{ll}
Y \in refusals(\mathsf{ioProcess}\,(ioC)) & \\
\quad \Leftrightarrow \forall\, inI;\; outI : Value \mid & [\mathrm{def}^{\,n} \text{ of } refusals(\mathsf{ioProcess}\,(ioC))] \\
\quad\quad inI \in \mathrm{dom}\, ioInit\; ioC \bullet & \\
\quad\quad\quad \mathrm{init}.inI.outI \in Y \Rightarrow & \\
\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\; ioC (\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin Y & \\
\quad \Leftrightarrow \forall\, inI;\; outI : Value \mid & [\mathrm{dom}\, ioInit\; ioA = \mathrm{dom}\, ioInit\; ioC] \\
\quad\quad inI \in \mathrm{dom}\, ioInit\; ioA \bullet & \\
\quad\quad\quad \mathrm{init}.inI.outI \in Y \Rightarrow & \\
\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\; ioC (\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin Y &
\end{array}
$$

$$\begin{array}{ll}
\Rightarrow \forall\, inI;\ outI : Value \mid & [\mathrm{ran}\ ioInit\ ioC \subseteq \mathrm{ran}\ ioInit\ ioA] \\
\quad inI \in \mathrm{dom}\ ioInit\ ioA \bullet & \\
\quad\quad \mathrm{init}.inI.outI \in Y \Rightarrow & \\
\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioA(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin Y & \\
& \\
\Rightarrow Y \in refusals(\mathsf{ioProcess}\,(ioA)) & [\mathrm{def}^n \text{ of } refusals(\mathsf{ioProcess}\,(ioA))]
\end{array}$$

From this we can immediately see that the failures set at initialisation of process $\mathsf{ioProcess}\,(ioC)$ must be a subset of the failures set at initialisation of process $\mathsf{ioProcess}\,(ioA)$.

true

$$\begin{array}{ll}
\Leftrightarrow \forall\, Y : \mathbb{P}\,\Sigma \bullet & [\text{by above argument}] \\
\quad Y \in refusals(\mathsf{ioProcess}\,(ioC)) & \\
\quad\quad \Rightarrow Y \in refusals(\mathsf{ioProcess}\,(ioA)) & \\
& \\
\Leftrightarrow \forall\, Y : \mathbb{P}\,\Sigma \bullet & [refusals \text{ of ioProcess}] \\
\quad Y \in \{X : \mathbb{P}\,\Sigma \mid & \\
\quad\quad\quad \forall\, inI, outI : Value \mid inI \in ioInit\ ioC \bullet & \\
\quad\quad\quad\quad \mathrm{init}.inI.outI \in X \Rightarrow & \\
\quad\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioC(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X\} & \\
\quad\quad \Rightarrow Y \in \{X : \mathbb{P}\,\Sigma \mid & \\
\quad\quad\quad\quad \forall\, inI, outI : Value \mid inI \in ioInit\ ioA \bullet & \\
\quad\quad\quad\quad\quad \mathrm{init}.inI.outI \in X \Rightarrow & \\
\quad\quad\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioA(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X\} & \\
& \\
\Leftrightarrow \{X : \mathbb{P}\,\Sigma \mid & [\mathrm{def}^n \text{ of } \subseteq] \\
\quad \forall\, inI, outI : Value \bullet & \\
\quad\quad \mathrm{init}.inI.outI \in X \Rightarrow & \\
\quad\quad\quad outI \notin \mathrm{ran}(ioInit\ ioC(\!|\ \{inI\}\ |\!)) & \\
\quad\quad\quad \vee & \\
\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioC(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X\} & \\
\quad \subseteq & \\
\quad \{X : \mathbb{P}\,\Sigma \mid & \\
\quad\quad \forall\, inI, outI : Value \bullet & \\
\quad\quad\quad \mathrm{init}.inI.outI \in X \Rightarrow & \\
\quad\quad\quad\quad outI \notin \mathrm{ran}(ioInit\ ioA(\!|\ \{inI\}\ |\!)) & \\
\quad\quad\quad\quad \vee & \\
\quad\quad\quad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioA(\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X\} &
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow \{X : \mathbb{P}\,\Sigma \mid \\
\qquad \forall\, inI, outI : Value \bullet \\
\qquad\qquad \mathrm{init}.inI.outI \in X \Rightarrow \\
\qquad\qquad\qquad outI \notin \mathrm{ran}(ioInit\; ioC(\!|\; \{inI\} \;|\!)) \\
\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad \exists\, outI' : \mathrm{ran}(ioInit\; ioC(\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad \bullet\; (\;\langle\rangle\;, X)\} \\
\qquad \subseteq \\
\qquad \{X : \mathbb{P}\,\Sigma \mid \\
\qquad\qquad \forall\, inI, outI : Value \bullet \\
\qquad\qquad\qquad \mathrm{init}.inI.outI \in X \Rightarrow \\
\qquad\qquad\qquad\qquad outI \notin \mathrm{ran}(ioInit\; ioA(\!|\; \{inI\} \;|\!)) \\
\qquad\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad\qquad \exists\, outI' : \mathrm{ran}(ioInit\; ioA(\!|\; \{inI\} \;|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad\qquad \bullet\; (\;\langle\rangle\;, X)\}
\end{array}
\qquad [\text{set theory}]
$$

Failures after all finished traces

In this section we consider failures after all finished traces.

Recall that the failures for $\mathsf{ioProcess}\,(ioA)$ after all finished traces is given by

$$
\begin{array}{l}
\{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value;\; ins, outs : \mathrm{seq}\; Value \mid \\
\qquad \#ins = \#outs = n \\
\qquad \wedge \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad\qquad startBatch\; ioA \mathbin{;} runBatch\; ioA \langle op \rangle^n \mathbin{;} stopBatch\; ioA \\
\qquad\qquad \bullet\; (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \ldots n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\}
\end{array}
$$

We now present a lemma which states that the failures of $\mathsf{ioProcess}\,(ioC)$ after all finished traces is a subset of the failures of $\mathsf{ioProcess}\,(ioA)$ precisely when, for each natural number n, the relation representing the composition of non-totalised versions of $startBatch\; ioC$, $runBatch\; ioC\,\langle op \rangle^n$ and $stopBatch\; ioC$ is a subset of the relation representing the composition of non-totalised versions of $startBatch\; ioA$, $runBatch\; ioA\,\langle op \rangle^n$ and $stopBatch\; ioA$.

Lemma 10

Given ioData types ioA and ioC of type $ioData[Value, Name, Local]$,

$$
\begin{array}{l}
(\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad \#ins = \#outs = n \\
\quad \wedge \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad startBatch\ ioC \fatsemi runBatch\ ioC\langle op \rangle^n \fatsemi stopBatch\ ioC \\
\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} \\
\quad \subseteq \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad \#ins = \#outs = n \\
\qquad \wedge \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad\quad startBatch\ ioA \fatsemi runBatch\ ioA\langle op \rangle^n \fatsemi stopBatch\ ioA \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\}) \\
\Leftrightarrow \\
(\forall n : \mathbb{N} \bullet \\
\quad startBatch\ ioC \fatsemi runBatch\ ioC\langle op \rangle^n \fatsemi stopBatch\ ioC \\
\qquad \subseteq \\
\qquad startBatch\ ioA \fatsemi runBatch\ ioA\langle op \rangle^n \fatsemi stopBatch\ ioA)
\end{array}
$$

□

Proof

$$
\begin{array}{l}
\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad \#ins = \#outs = n \\
\quad \wedge \\
\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad startBatch\ ioC \fatsemi runBatch\ ioC\langle op \rangle^n \fatsemi stopBatch\ ioC \\
\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} \\
\quad \subseteq \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad \#ins = \#outs = n \\
\qquad \wedge \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad\quad startBatch\ ioA \fatsemi runBatch\ ioA\langle op \rangle^n \fatsemi stopBatch\ ioA \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\}
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value; & \text{[property of } \subseteq\text{]} \\
\qquad\qquad\qquad ins, outs : \mathrm{seq}\; Value \mid & \\
\quad \#ins = \#outs = n \bullet & \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in & \\
\qquad\qquad startBatch\; ioC \fatsemi runBatch\; ioC \langle op \rangle^n \fatsemi stopBatch\; ioC & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in & \\
\qquad\qquad\quad startBatch\; ioA \fatsemi runBatch\; ioA \langle op \rangle^n \fatsemi stopBatch\; ioA &
\end{array}$$

$$\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet & \text{[property of } \subseteq\text{]} \\
\quad startBatch\; ioC \fatsemi runBatch\; ioC \langle op \rangle^n \fatsemi stopBatch\; ioC & \\
\qquad \subseteq & \\
\qquad startBatch\; ioA \fatsemi runBatch\; ioA \langle op \rangle^n \fatsemi stopBatch\; ioA &
\end{array}$$

Failures after all unfinished traces

In this section we consider failures after all unfinished traces.

Recall that the failures set for $ioProcess\, ioA$ after all unfinished traces is given by:

$$\begin{array}{l}
\{ n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \mathrm{seq}\; Value \mid \\
\quad (\,\#ins = \#outs = n \\
\quad\; \wedge \\
\quad\; \exists\, a : Local \bullet \\
\qquad \langle inI \rangle \frown ins \mapsto (a,\; \langle\rangle\;, \langle outI \rangle \frown outs) \in \\
\qquad\quad startBatch\; ioA \fatsemi runBatch\; ioA \langle op \rangle^n \\
\qquad \wedge \\
\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\, ioOps\;\; ioA\; op \bullet \\
\qquad\quad \mathrm{op}.in.out \in X \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\; ioA\; op (\!|\; \{(a, in)\}\; |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \wedge \\
\qquad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\, ioFinal\; ioA \bullet \\
\qquad\quad \mathrm{final}.inF.outF \in X \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\; ioA\; op (\!|\; \{(a, in)\}\; |\!)) \bullet \mathrm{final}.inF.outF' \notin X\;) \\
\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle, X) \}
\end{array}$$

Due to the complexity of this set we will go into very detailed explanations and justifications at each step whilst manipulating the predicate which says that the failures of $\mathsf{ioProcess}\,(ioC)$ after all unfinished traces is a subset of the failures of $\mathsf{ioProcess}\,(ioA)$ after all unfinished traces into an equivalent but more manageable form.

Our objective for this lemma is to express the predicate as an equivalent predicate which is not defined in terms of sets of events.

Lemma 11
Given ioData types ioA and ioC of type $ioData[Value, Name, Local]$

$$
\begin{array}{l}
(\,\{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \mathrm{seq}\, Value \mid \\
\quad (\,\#ins = \#outs = n \\
\quad\;\; \wedge \\
\quad\;\; \exists\, c : Local \bullet \\
\qquad\quad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad startBatch\ ioC \fatsemi runBatch\ ioC \langle op \rangle^n \\
\qquad\quad \wedge \\
\qquad\quad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad out \notin \mathrm{ran}(ioOps\ ioC\ op(\!|\ \{(c, in)\}\ |\!)) \\
\qquad\qquad \vee \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op(\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\quad \wedge \\
\qquad\quad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad outF \notin \mathrm{ran}(ioFinal\ ioC(\!|\ \{(c, inF)\}\ |\!)) \\
\qquad\qquad \vee \\
\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op(\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\,) \\
\qquad\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\} \\
\quad \subseteq \\
\quad \{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \mathrm{seq}\, Value \mid \\
\qquad (\,\#ins = \#outs = n \\
\qquad\;\; \wedge \\
\qquad\;\; \exists\, a : Local \bullet \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad\quad startBatch\ ioA \fatsemi runBatch\ ioA \langle op \rangle^n \\
\qquad\qquad \wedge \\
\qquad\qquad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\quad out \notin \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \\
\qquad\qquad\quad \vee \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\qquad \wedge \\
\qquad\qquad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\quad outF \notin \mathrm{ran}(ioFinal\ ioA(\!|\ \{(a, inF)\}\ |\!)) \\
\qquad\qquad\quad \vee \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\,) \\
\qquad\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}\,)
\end{array}
$$

$$
\begin{aligned}
\Leftrightarrow (\forall\, n : \mathbb{N} \bullet \\
&\quad startBatch\ ioC \fatsemi runBatch\ ioC\ \langle op \rangle^n \fatsemi stopBatch\ ioC \\
&\qquad \subseteq \\
&\qquad startBatch\ ioA \fatsemi runBatch\ ioA\ \langle op \rangle^n \fatsemi stopBatch\ ioA \\
\wedge \\
\forall\, n : \mathbb{N};\ inI, outI, in : Value;\ ins, outs' : \mathrm{seq}\ Value \mid \\
&\quad \#ins = n \wedge \#outs' = n \bullet \\
&\qquad \exists\, c : ioState\ ioC \bullet \\
&\qquad\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
&\qquad\qquad\quad startBatch\ ioC \fatsemi runBatch\ ioC\ (\langle op \rangle^n) \\
&\qquad\qquad \wedge \\
&\qquad\qquad (c, in) \notin \mathrm{dom}\ ioOps\ ioC\ op \\
&\qquad\quad \Rightarrow \\
&\qquad\quad \exists\, a : ioState\ ioA \bullet \\
&\qquad\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
&\qquad\qquad\qquad startBatch\ ioA \fatsemi runBatch\ ioA\ (\langle op \rangle^n) \\
&\qquad\qquad\quad \wedge \\
&\qquad\qquad\quad (a, in) \notin \mathrm{dom}\ ioOps\ ioC\ op \\
\wedge \\
\forall\, n : \mathbb{N};\ inI, outI, inF : Value;\ ins, outs' : \mathrm{seq}\ Value \mid \\
&\quad \#ins = n \wedge \#outs' = n \bullet \\
&\qquad \exists\, c : ioState\ ioC \bullet \\
&\qquad\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
&\qquad\qquad\quad startBatch\ ioC \fatsemi runBatch\ ioC\ (\langle op \rangle^n) \\
&\qquad\qquad \wedge \\
&\qquad\qquad (c, inF) \notin \mathrm{dom}\ ioFinal\ ioC \\
&\qquad\quad \Rightarrow \\
&\qquad\quad \exists\, a : ioState\ ioA \bullet \\
&\qquad\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
&\qquad\qquad\qquad startBatch\ ioA \fatsemi runBatch\ ioA\ (\langle op \rangle^n) \\
&\qquad\qquad\quad \wedge \\
&\qquad\qquad\quad (a, inF) \notin \mathrm{dom}\ ioFinal\ ioA\ )
\end{aligned}
$$

□

Proof

Simplifying the failures set

Before even beginning to consider failures refinement after all unfinished traces we simplify this failures set a little.

First consider the following predicate constraining which events are in the refusals set.

$$
\begin{array}{l}
\{\, X : \mathbb{P}\,\Sigma;\ \ldots \mid \\
\quad \vdots \\
\quad \exists\, a : \mathit{Local} \bullet \\
\qquad \vdots \\
\qquad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \bullet \\
\qquad\quad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\
\qquad\qquad (\, \mathit{out} \notin \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!)) \\
\qquad\qquad\ \ \vee \\
\qquad\qquad\ \ \exists\, \mathit{out}' : \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X) \\
\qquad \vdots \\
\qquad\quad \bullet\, (\ldots, X)\, \}
\end{array}
$$

First observe that for all *in* such that (a, in) lies in the domain of *ioOps ioA op*, the clause

$$\mathit{out} \notin \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!))$$

will automatically be true for all values of *out*. So, our predicate may be simplified to

$$
\begin{array}{l}
\{\, X : \mathbb{P}\,\Sigma;\ \ldots \mid \\
\quad \vdots \\
\quad \exists\, a : \mathit{Local} \mid \\
\qquad \vdots \\
\qquad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \mid (a, \mathit{in}) \in \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \bullet \\
\qquad\quad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\
\qquad\qquad (\, \mathit{out} \notin \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!)) \\
\qquad\qquad\ \ \vee \\
\qquad\qquad\ \ \exists\, \mathit{out}' : \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X) \\
\qquad \vdots \\
\qquad\quad \bullet\, (\ldots, X)\, \}
\end{array}
$$

Furthermore, since (a, in) lies in the domain of *ioOps ioA op* we will always be able to find some value out_a which lies in the range of the relational image of (a, in) under *ioOps ioA op*. We may then choose a set X containing the event $\mathrm{op}.in.out_a$ and hence since the clause

$$\mathit{out}_a \notin \mathrm{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op}(\!|\ \{(a, \mathit{in})\}\ |\!))$$

is false, it must be the case that for any $in : Value$ we can find a value out' from the relational image such that the event $\mathrm{op}.in.out'$ is not in the set X.

It follows that our predicate may therefore be reduced further:

$$
\begin{array}{l}
\{\, X : \mathbb{P}\,\Sigma;\ \ldots \mid \\
\quad \vdots \\
\quad \exists\, a : Local \mid \\
\qquad \vdots \\
\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\, ioOps\ ioA\ op \bullet \\
\qquad\quad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \vdots \\
\qquad\quad \bullet\, (\ldots, X)\,\}
\end{array}
$$

We may apply similar arguments to the predicate constraining which $\mathrm{final}.inF.outF$ events lie in our refusals set and thereby simplify our failures set to the following form:

$$
\begin{array}{l}
\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad (\,\#ins = \#outs = n \\
\quad\ \wedge \\
\quad\ \exists\, a : Local \bullet \\
\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\quad startBatch\ ioA \fatsemi runBatch\ ioA \langle op \rangle^{n} \\
\qquad \wedge \\
\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\, ioOps\ ioA\ op \bullet \\
\qquad\quad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \wedge \\
\qquad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\, ioFinal\ ioA \bullet \\
\qquad\quad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) \\
\qquad\quad \bullet\, (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}
\end{array}
$$

Reasoning about failures refinement

We know that in order for $\mathsf{ioProcess}\,(ioA)$ to be failures refined by $\mathsf{ioProcess}\,(ioC)$, the failures set of $\mathsf{ioProcess}\,(ioC)$ after an unfinished trace must be a subset of the failures set of $\mathsf{ioProcess}\,(ioA)$ after an unfinished trace.

We may express this condition in terms of our simplified sets above and then reason as follows:

$$
\begin{array}{l}
\{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad (\ \#ins = \#outs = n \\
\quad\ \wedge \\
\quad\ \exists\, c : Local \bullet \\
\qquad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\quad startBatch\ ioC \fatsemi runBatch\ ioC \langle op \rangle^{n} \\
\qquad \wedge \\
\qquad \forall\, in, out : Value \mid (c, in) \in \mathrm{dom}\ ioOps\ ioC\ op \bullet \\
\qquad\quad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \wedge \\
\qquad \forall\, inF, outF : Value \mid (c, inF) \in \mathrm{dom}\ ioFinal\ ioC \bullet \\
\qquad\quad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \ldots n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\} \\
\quad \subseteq \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad (\ \#ins = \#outs = n \\
\qquad \wedge \\
\qquad \exists\, a : Local \bullet \\
\qquad\quad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad startBatch\ ioA \fatsemi runBatch\ ioA \langle op \rangle^{n} \\
\qquad\quad \wedge \\
\qquad\quad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\ ioOps\ ioA\ op \bullet \\
\qquad\qquad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\quad \wedge \\
\qquad\quad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) \\
\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \ldots n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \qquad \text{[set theory]} \\
\quad (\#ins = \#outs = n \\
\quad \wedge \\
\quad \exists\, c : Local \bullet \\
\qquad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\quad startBatch\ ioC \circ runBatch\ ioC \langle op \rangle^{n} \\
\qquad \wedge \\
\qquad \forall\, in, out : Value \mid (c, in) \in \mathrm{dom}\ ioOps\ ioC\ op \bullet \\
\qquad\quad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad \wedge \\
\qquad \forall\, inF, outF : Value \mid (c, inF) \in \mathrm{dom}\ ioFinal\ ioC \bullet \\
\qquad\quad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ )\} \\
\quad \subseteq \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\qquad (\#ins = \#outs = n \\
\qquad \wedge \\
\qquad \exists\, a : Local \bullet \\
\qquad\quad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad startBatch\ ioA \circ runBatch\ ioA \langle op \rangle^{n} \\
\qquad\quad \wedge \\
\qquad\quad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\ ioOps\ ioA\ op \bullet \\
\qquad\qquad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\quad \wedge \\
\qquad\quad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ )\}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid & \text{[set theory]} \\
\quad \#ins = \#outs = n \bullet & \\
\quad\quad (\, \exists\, c : Local \bullet & \\
\quad\quad\quad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in & \\
\quad\quad\quad\quad startBatch\ ioC \mathbin{⨾} runBatch\ ioC \langle op \rangle^n & \\
\quad\quad\quad \wedge & \\
\quad\quad\quad \forall\, in, out : Value \mid (c, in) \in \mathrm{dom}\ ioOps\ ioC\ op \bullet & \\
\quad\quad\quad\quad \mathrm{op}.in.out \in X \Rightarrow & \\
\quad\quad\quad\quad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op \llparenthesis\ \{(c, in)\}\ \rrparenthesis) \bullet \mathrm{op}.in.out' \notin X & \\
\quad\quad\quad \wedge & \\
\quad\quad\quad \forall\, inF, outF : Value \mid (c, inF) \in \mathrm{dom}\ ioFinal\ ioC \bullet & \\
\quad\quad\quad\quad \mathrm{final}.inF.outF \in X \Rightarrow & \\
\quad\quad\quad\quad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op \llparenthesis\ \{(c, in)\}\ \rrparenthesis) \bullet \mathrm{final}.inF.outF' \notin X\ ) & \\
\quad\quad\quad \Rightarrow & \\
\quad\quad\quad (\, \exists\, a : Local \bullet & \\
\quad\quad\quad\quad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in & \\
\quad\quad\quad\quad\quad startBatch\ ioA \mathbin{⨾} runBatch\ ioA \langle op \rangle^n & \\
\quad\quad\quad\quad \wedge & \\
\quad\quad\quad\quad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\ ioOps\ ioA\ op \bullet & \\
\quad\quad\quad\quad\quad \mathrm{op}.in.out \in X \Rightarrow & \\
\quad\quad\quad\quad\quad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op \llparenthesis\ \{(a, in)\}\ \rrparenthesis) \bullet \mathrm{op}.in.out' \notin X & \\
\quad\quad\quad\quad \wedge & \\
\quad\quad\quad\quad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\ ioFinal\ ioA \bullet & \\
\quad\quad\quad\quad\quad \mathrm{final}.inF.outF \in X \Rightarrow & \\
\quad\quad\quad\quad\quad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op \llparenthesis\ \{(a, in)\}\ \rrparenthesis) \bullet \mathrm{final}.inF.outF' \notin X\ ) &
\end{array}
$$

Now, since the set of events X may contain just the single event $\mathrm{op}.in.out$ or the single event $\mathrm{final}.inF.outF$ for all possible values of $in, out, inF, outF$, we may separate out the clauses to get the following equivalent predicate:

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \mathrm{seq}\; Value \mid \\
\qquad \#ins = \#outs = n \bullet \\
\qquad\qquad (\, \exists\, c : Local \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad\qquad startBatch\; ioC \mathbin{⨾} runBatch\; ioC \langle op \rangle^{n} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \forall\, in, out : Value \mid (c, in) \in \mathrm{dom}\; ioOps\; ioC\; op \bullet \\
\qquad\qquad\qquad\qquad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\qquad\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\; ioC\; op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X) \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\, \exists\, a : Local \bullet \\
\qquad\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad\qquad\qquad startBatch\; ioA \mathbin{⨾} runBatch\; ioA \langle op \rangle^{n} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\; ioOps\; ioA\; op \bullet \\
\qquad\qquad\qquad\qquad\qquad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\; ioA\; op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \mathrm{seq}\; Value \mid \\
\qquad \#ins = \#outs = n \bullet \\
\qquad\qquad (\, \exists\, c : Local \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad\qquad startBatch\; ioC \mathbin{⨾} runBatch\; ioC \langle op \rangle^{n} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \forall\, inF, outF : Value \mid (c, inF) \in \mathrm{dom}\; ioFinal\; ioC \bullet \\
\qquad\qquad\qquad\qquad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\qquad\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\; ioC\; op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\, \exists\, a : Local \bullet \\
\qquad\qquad\qquad\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad\qquad\qquad startBatch\; ioA \mathbin{⨾} runBatch\; ioA \langle op \rangle^{n} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\; ioFinal\; ioA \bullet \\
\qquad\qquad\qquad\qquad\qquad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\; ioA\; op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ )
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \qquad \text{[property of } \mid\text{]} \\
\quad \#ins = \#outs = n \bullet \\
\qquad (\,\exists\, c : Local \mid \langle inI\rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\qquad\qquad\qquad startBatch\ ioC \mathbin{;} runBatch\ ioC\langle op\rangle^{n} \bullet \\
\qquad\quad \forall\, in, out : Value \mid (c, in) \in \mathrm{dom}\ ioOps\ ioC\ op \bullet \\
\qquad\qquad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op(\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X) \\
\qquad\quad \Rightarrow \\
\qquad\quad (\,\exists\, a : Local \mid \langle inI\rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\qquad\qquad\qquad startBatch\ ioA \mathbin{;} runBatch\ ioA\langle op\rangle^{n} \bullet \\
\qquad\qquad \forall\, in, out : Value \mid (a, in) \in \mathrm{dom}\ ioOps\ ioA\ op \bullet \\
\qquad\qquad\quad \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\qquad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X) \\
\ \ \wedge \\
\ \ \forall\, n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid \\
\quad \#ins = \#outs = n \bullet \\
\qquad (\,\exists\, c : Local \mid \langle inI\rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\qquad\qquad\qquad startBatch\ ioC \mathbin{;} runBatch\ ioC\langle op\rangle^{n} \bullet \\
\qquad\quad \forall\, inF, outF : Value \mid (c, inF) \in \mathrm{dom}\ ioFinal\ ioC \bullet \\
\qquad\qquad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op(\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) \\
\qquad\quad \Rightarrow \\
\qquad\quad (\,\exists\, a : Local \mid \langle inI\rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\qquad\qquad\qquad startBatch\ ioA \mathbin{;} runBatch\ ioA\langle op\rangle^{n} \bullet \\
\qquad\qquad \forall\, inF, outF : Value \mid (a, inF) \in \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\quad \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\qquad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op(\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ )
\end{array}
$$

There is a vast amount of information in this predicate. Our next goal is to express it without referring to the set of events X.

Inputs which lie in the domain

Our first observation is that the predicate is defined in terms of all possible sets of events. Hence for any given local state $c : ioState\ ioC$ such that

$$
\begin{array}{l}
\langle inI\rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI\rangle \frown outs) \in \\
\quad startBatch\ ioC \mathbin{;} runBatch\ ioC\langle op\rangle^{n}
\end{array}
$$

and each value in such that $(c, in) \in \mathrm{dom}\ ioOps\ ioC\ op$ we may choose separately each value out' such that out' lies in the relational image of (c, in) under $ioOps\ ioC\ op$. We may then set X to be

$$
X \ =\ \bigcup_{in\, :\, Value} \{out : Value \mid out \neq out' \bullet \mathrm{op}.in.out\}
$$

In this case, the clause

$$\begin{aligned}
&\exists\, c : \mathit{Local} \;\mid\; \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
&\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC} \langle \mathit{op} \rangle^n \;\bullet \\
&\quad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \mid (c, \mathit{in}) \in \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \bullet \\
&\qquad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\
&\qquad\quad \exists\, \mathit{out}' : \operatorname{ran}(\mathit{ioOps}\ \mathit{ioC}\ \mathit{op} (\!|\ \{(c, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X
\end{aligned}$$

must be true and hence it follows from the implication that the following clause must also be true

$$\begin{aligned}
&\exists\, a : \mathit{Local} \;\mid\; \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (a,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
&\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA} \langle \mathit{op} \rangle^n \;\bullet \\
&\quad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \mid (a, \mathit{in}) \in \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \bullet \\
&\qquad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\
&\qquad\quad \exists\, \mathit{out}' : \operatorname{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op} (\!|\ \{(a, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X
\end{aligned}$$

Since we can apply this to all such out' it follows that for each local state $c : \mathit{ioState}\ \mathit{ioC}$ satisfying the constraints

$$\begin{aligned}
&\langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
&\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC} \langle \mathit{op} \rangle^n
\end{aligned}$$

we can find a local state $a : \mathit{ioState}\ \mathit{ioA}$ satisfying the constraints

$$\begin{aligned}
&\langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (a,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
&\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA} \langle \mathit{op} \rangle^n
\end{aligned}$$

such that for each value in satisfying the constraint

$$(c, \mathit{in}) \in \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op}$$

the following must hold:

$$\operatorname{ran} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} (\!|\ \{c, \mathit{in}\}\ |\!) \;\subseteq\; \operatorname{ran} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} (\!|\ \{a, \mathit{in}\}\ |\!)$$

Applying a similar argument to the clause concerning final events we obtain the following result:

For each local state $c : \mathit{ioState}\ \mathit{ioC}$ satisfying the constraints

$$\begin{aligned}
&\langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
&\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC} \langle \mathit{op} \rangle^n
\end{aligned}$$

we can find a local state $a : \mathit{ioState}\ \mathit{ioA}$ satisfying the constraints

$$\begin{array}{l} \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (a,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\ \qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨟} \mathit{runBatch}\ \mathit{ioA} \langle \mathit{op} \rangle^n \end{array}$$

such that for each value inF satisfying the constraint

$$(c, \mathit{inF}) \in \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC}$$

the following must hold:

$$\mathit{ioFinal}\ \mathit{ioC} (\!|\ \{c, \mathit{inF}\}\ |\!) \ \subseteq\ \mathit{ioFinal}\ \mathit{ioA} (\!|\ \{a, \mathit{inF}\}\ |\!)$$

All this combined together gives us the following very simple result:

$$\begin{array}{l} \forall\, n : \mathbb{N} \\ \qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨟} \mathit{runBatch}\ \mathit{ioC} \mathbin{⨟} \mathit{stopBatch}\ \mathit{ioC} \\ \qquad\qquad \subseteq \\ \qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨟} \mathit{runBatch}\ \mathit{ioA} \mathbin{⨟} \mathit{stopBatch}\ \mathit{ioA} \end{array}$$

Inputs which do not lie in the domain

There is more information contained in our predicate.

Once more we call upon the fact that the predicate is defined in terms of all possible sets of events. Hence, for any given local state $c : \mathit{ioState}\ \mathit{ioC}$ such that

$$\begin{array}{l} \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\ \qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨟} \mathit{runBatch}\ \mathit{ioC} \langle \mathit{op} \rangle^n \end{array}$$

we may set the set X to be

$$X \ = \ \{\mathit{in}, \mathit{out} : \mathit{Value} \mid (c, \mathit{in}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \bullet \mathrm{op}.\mathit{in}.\mathit{out}\}$$

It then follows that the following clause

$$\begin{array}{l} \exists\, c : \mathit{Local} \ \mid\ \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\ \qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨟} \mathit{runBatch}\ \mathit{ioC} \langle \mathit{op} \rangle^n \ \bullet \\ \qquad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \mid (c, \mathit{in}) \in \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \bullet \\ \qquad\qquad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\ \qquad\qquad\qquad \exists\, \mathit{out}' : \operatorname{ran}(\mathit{ioOps}\ \mathit{ioC}\ \mathit{op} (\!|\ \{(c, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X \end{array}$$

must trivially be true, and hence it follows from the implication that the following clause must also be true.

$$
\begin{array}{l}
\exists\, a : \mathit{Local} \mid \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (a,\ \langle\rangle\ , \langle \mathit{outI} \rangle \frown \mathit{outs}) \in \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA} \langle \mathit{op} \rangle^n \ \bullet \\
\quad \forall\, \mathit{in}, \mathit{out} : \mathit{Value} \mid (a, \mathit{in}) \in \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \bullet \\
\qquad \mathrm{op}.\mathit{in}.\mathit{out} \in X \Rightarrow \\
\qquad\quad \exists\, \mathit{out}' : \operatorname{ran}(\mathit{ioOps}\ \mathit{ioA}\ \mathit{op} (\!|\ \{(a, \mathit{in})\}\ |\!)) \bullet \mathrm{op}.\mathit{in}.\mathit{out}' \notin X
\end{array}
$$

This will only be the case if $(a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}$

This gives us the result that

$$
\begin{array}{l}
\exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathit{op} \rangle^n) \\
\qquad \wedge \\
\qquad (c, \mathit{in}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\quad \Rightarrow \\
\quad \exists\, a : \mathit{ioState}\ \mathit{ioA} \bullet \\
\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathit{op} \rangle^n) \\
\qquad\quad \wedge \\
\qquad\quad (a, \mathit{in}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op}
\end{array}
$$

We may apply a similar argument to the clause concerning final.$\mathit{inF}.\mathit{outF}$ events and obtain the following result:

$$
\begin{array}{l}
\exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathit{op} \rangle^n) \\
\qquad \wedge \\
\qquad (c, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC} \\
\quad \Rightarrow \\
\quad \exists\, a : \mathit{ioState}\ \mathit{ioA} \bullet \\
\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathit{op} \rangle^n) \\
\qquad\quad \wedge \\
\qquad\quad (a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

Combining our observations

We are finally able to express our predicate *not* in terms of the set of events X as follows:

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ \langle \mathit{op} \rangle^{n} \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ \langle \mathit{op} \rangle^{n} \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n \wedge \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathit{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists\, a : \mathit{ioState}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathit{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n \wedge \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathit{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists\, a : \mathit{ioState}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathit{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

Refinement of ioData types is equivalent to refinement of their corresponding ioProcesses

In this section we finally prove Theorem 5 that the refinement of ioData types is equivalent to refinement of their corresponding ioProcesses. That is, given two ioData types, the first is refined by the second precisely when the ioProcess corresponding to the first is refined by the ioProcess corresponding to the second.

This holds for all ioData types which have the same domain for their initialisation operations and such that the range of the relational image of any point in that domain under the initialisation operation of the second is a subset of the range of the relational image of that same point under the initialisation operation of the first.

Theorem 5 Given generic parameters $Value, Name, local$,

$$\begin{array}{l} \forall\, ioA;\ ioC : ioData[Value, Name, Local] \bullet \\ \quad (ioA \sqsubseteq ioC) \;\Leftrightarrow\; (\mathsf{ioProcess}\,(ioA) \sqsubseteq_F \mathsf{ioProcess}\,(ioC)) \end{array}$$

Proof

$$ioA \sqsubseteq ioC$$

$$\begin{array}{ll} \Leftrightarrow \forall\, n : \mathbb{N} \bullet & [\text{def}^{\,n} \text{ of } ioSemantics] \\ \quad ioSemantics\ ioC\,(\langle \text{op} \rangle^n) \subseteq ioSemantics\ ioA\,(\langle \text{op} \rangle^n) & \end{array}$$

$$\begin{array}{ll} \Leftrightarrow \forall\, n : \mathbb{N} \bullet & [\text{def}^{\,n} \text{ of } ioSemantics] \\ \quad startBatch^{\top}\ ioC \fatsemi runBatch^{\top}\ ioC\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioC & \\ \qquad \subseteq & \\ \qquad startBatch^{\top}\ ioA \fatsemi runBatch^{\top}\ ioA\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioA & \end{array}$$

$$\begin{array}{ll} \Leftrightarrow \forall\, n : \mathbb{N} \bullet & [\text{set theory}] \\ \quad startBatch^{\top}\ ioC \fatsemi runBatch^{\top}\ ioC\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioC & \\ \qquad\qquad \rhd \{outs' : \text{seq}\ Value \mid \#outs' = n + 2\} & \\ \qquad \subseteq & \\ \qquad startBatch^{\top}\ ioA \fatsemi runBatch^{\top}\ ioA\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioA & \\ \qquad\qquad\quad \rhd \{outs' : \text{seq}\ Value \mid \#outs' = n + 2\} & \\ \quad \wedge & \\ \quad startBatch^{\top}\ ioC \fatsemi runBatch^{\top}\ ioC\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioC & \\ \qquad\qquad \rhd \{outs' : \text{seq}\ Value \mid \#outs' < n + 2\} & \\ \qquad \subseteq & \\ \qquad startBatch^{\top}\ ioA \fatsemi runBatch^{\top}\ ioA\,(\langle \text{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioA & \\ \qquad\qquad\quad \rhd \{outs' : \text{seq}\ Value \mid \#outs' < n + 2\} & \end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet & [\text{def}^{\,n} \text{ of } \_^{\top}] \\
\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} & \\
\quad \wedge & \\
\quad \mathit{startBatch}^{\top}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} & \\
\qquad\qquad \rhd \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2 \} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} & \\
\qquad\qquad\quad \rhd \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2 \} & \\
& \\
\Leftrightarrow \forall\, n : \mathbb{N} \bullet & [\text{property of } \forall] \\
\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} & \\
\ \ \wedge & \\
\ \ \forall\, n : \mathbb{N} \bullet & \\
\quad \mathit{startBatch}^{\top}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} & \\
\qquad\qquad \rhd \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2 \} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} & \\
\qquad\qquad\quad \rhd \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2 \} &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill \text{[set theory]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} \\
\qquad \wedge \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill \text{[definition of } \mathit{startBatch}^{\top} \text{ etc.]} \\
\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad \subseteq \\
\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\wedge \\
\forall n : \mathbb{N} \bullet \\
\quad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \operatorname{seq} \mathit{Value} \mid \\
\qquad\quad \mathit{inI} \notin \operatorname{dom} \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle \ \} \\
\qquad \subseteq \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \operatorname{seq} \mathit{Value} \mid \\
\qquad\qquad \mathit{inI} \notin \operatorname{dom} \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle \ \} \\
\quad \wedge \\
\quad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \operatorname{seq} \mathit{Value} \mid \mathit{inI} \in \operatorname{dom} \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad \lhd \\
\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\quad \rhd \\
\qquad\quad \{ \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} \\
\qquad \subseteq \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \operatorname{seq} \mathit{Value} \mid \mathit{inI} \in \operatorname{dom} \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} \\
\qquad\quad \lhd \\
\qquad\quad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad \rhd \\
\qquad\qquad \{ \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \# \mathit{outs}' < n + 2 \}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet & \text{[property of } \forall\text{]} \\
\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} & \\
\wedge & \\
\forall\, n : \mathbb{N} \bullet & \\
\quad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\quad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle\ \} & \\
\qquad \subseteq & \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle\ \} & \\
\wedge & \\
\forall\, n : \mathbb{N} \bullet & \\
\quad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} & \\
\qquad \lhd & \\
\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} & \\
\qquad\quad \rhd & \\
\qquad\quad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} & \\
\qquad \subseteq & \\
\qquad \{ \mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \} & \\
\qquad\quad \lhd & \\
\qquad\quad \mathit{startBatch}^{\top}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} & \\
\qquad\qquad \rhd & \\
\qquad\qquad \{ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \# \mathit{outs}' < n + 2 \} &
\end{array}
$$

$$\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad \text{[set theory]}\\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC}\\
\qquad\qquad \subseteq\\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA}\\
\quad \wedge\\
\quad \forall\, n : \mathbb{N} \bullet\\
\qquad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}\}\\
\qquad\qquad \subseteq\\
\qquad\qquad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA}\}\\
\quad \wedge\\
\quad \forall\, n : \mathbb{N} \bullet\\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\}\\
\qquad\qquad \lhd\\
\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC}\\
\qquad\qquad\qquad \rhd\\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}\\
\qquad\qquad \subseteq\\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\}\\
\qquad\qquad\qquad \lhd\\
\qquad\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA}\\
\qquad\qquad\qquad\qquad \rhd\\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}$$

$$\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad \text{[property of } \forall\text{]}\\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC}\\
\qquad\qquad \subseteq\\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA}\\
\quad \wedge\\
\quad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}\}\\
\qquad \subseteq\\
\qquad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA}\}\\
\quad \wedge\\
\quad \forall\, n : \mathbb{N} \bullet\\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\}\\
\qquad\qquad \lhd\\
\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC}\\
\qquad\qquad\qquad \rhd\\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}\\
\qquad\qquad \subseteq\\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\}\\
\qquad\qquad\qquad \lhd\\
\qquad\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA}\\
\qquad\qquad\qquad\qquad \rhd\\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} = \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}] \\
\quad\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\quad\quad\quad \subseteq \\
\quad\quad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}\} \\
\quad\quad \subseteq \\
\quad\quad \{\mathit{inI} : \mathit{Value} \mid \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}\} \\
\quad \land \\
\quad \forall\, n : \mathbb{N} \bullet \\
\quad\quad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\quad\quad\quad \lhd \\
\quad\quad\quad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\quad\quad\quad\quad \rhd \\
\quad\quad\quad\quad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\quad\quad\quad \subseteq \\
\quad\quad\quad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\quad\quad\quad\quad \lhd \\
\quad\quad\quad\quad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\quad\quad\quad\quad\quad \rhd \\
\quad\quad\quad\quad\quad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\text{set theory}] \\
\quad\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\quad\quad\quad \subseteq \\
\quad\quad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \mathrm{true} \\
\quad \land \\
\quad \forall\, n : \mathbb{N} \bullet \\
\quad\quad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\quad\quad\quad \lhd \\
\quad\quad\quad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\quad\quad\quad\quad \rhd \\
\quad\quad\quad\quad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\quad\quad\quad \subseteq \\
\quad\quad\quad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\quad\quad\quad\quad \lhd \\
\quad\quad\quad\quad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\quad\quad\quad\quad\quad \rhd \\
\quad\quad\quad\quad\quad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill [\text{property of } \land] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill [\mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} = \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle\ \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \mathit{inI} \notin \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle\rangle\ \} \\
\qquad \land \\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^{\top}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill \text{[property of } \subseteq \text{]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad \mathrm{true} \\
\qquad \wedge \\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioC} \fatsemi \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}^\top\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioA} \fatsemi \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}^\top\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \hfill \text{[property of } \wedge \text{]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioC} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad \lhd \\
\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioC} \fatsemi \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}^\top\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \{\mathit{inI} : \mathit{Value};\ \mathit{ins} : \mathrm{seq}\ \mathit{Value} \mid \mathit{inI} \in \mathrm{dom}\ \mathit{ioInit}\ \mathit{ioA} \bullet \langle \mathit{inI} \rangle \frown \mathit{ins}\} \\
\qquad\qquad\qquad \lhd \\
\qquad\qquad\qquad \mathit{startBatch}^\top\ \mathit{ioA} \fatsemi \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}^\top\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{\mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \#\mathit{outs}' < n + 2\}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\mathrm{def}^{n} \text{ of } \mathit{startBatch}^{\top}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n + 1 \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\qquad \#\mathit{outs}' < n + 1 \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\text{property of } \rhd] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad \wedge \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' = n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\qquad \#\mathit{outs}' = n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet & \text{[property of } \forall\text{]} \\
\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} & \\
\wedge & \\
\forall\, n : \mathbb{N} \bullet & \\
\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} & \\
\qquad\qquad \rhd & \\
\qquad\qquad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} & \\
\qquad\qquad\quad \rhd & \\
\qquad\qquad\quad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad\quad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} & \\
\wedge & \\
\forall\, n : \mathbb{N} \bullet & \\
\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} & \\
\qquad\qquad \rhd & \\
\qquad\qquad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad \#\mathit{outs}' = n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} & \\
\qquad \subseteq & \\
\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} & \\
\qquad\qquad\quad \rhd & \\
\qquad\qquad\quad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid & \\
\qquad\qquad\qquad\quad \#\mathit{outs}' = n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad [\text{def}^{\,n} \text{ of } \subseteq] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\quad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n + 1 \ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad [\mathrm{def}^{\,n}\mathrm{s\ of}\ \fatsemi,\ \mathit{runBatch}^{\top}\ \mathrm{and}\ \mathit{stopBatch}^{\top}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{ \mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\quad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}' \} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n + 1\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (c, \langle \mathit{last}\ \mathit{ins} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad (c, \mathit{last}\ \mathit{ins}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto (a, \langle \mathit{last}\ \mathit{ins} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (a, \mathit{last}\ \mathit{ins}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall n : \mathbb{N} \bullet \qquad\qquad\qquad\qquad\qquad [\text{substitution: } ins = is \frown \langle inF \rangle] \\
\qquad startBatch\ ioC \fatsemi runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \fatsemi stopBatch\ ioC \\
\qquad\qquad \subseteq \\
\qquad\qquad startBatch\ ioA \fatsemi runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \fatsemi stopBatch\ ioA \\
\quad \wedge \\
\quad \forall n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \fatsemi runBatch^{\top}\ ioC\ (\langle \mathrm{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioC \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ outI : Value;\ outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad \# outs' < n \bullet \langle outI \rangle \frown outs' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad startBatch\ ioA \fatsemi runBatch^{\top}\ ioA\ (\langle \mathrm{op} \rangle^n) \fatsemi stopBatch^{\top}\ ioA \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{ outI : Value;\ outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad\quad \# outs' < n \bullet \langle outI \rangle \frown outs' \} \\
\quad \wedge \\
\quad \forall n : \mathbb{N};\ inI, outI, inF : Value;\ is, outs' : \mathrm{seq}\ Value \mid \\
\qquad \#(is \frown \langle inF \rangle) = n + 1 \ \wedge\ \# outs' = n \bullet \\
\qquad\qquad \exists c : ioState\ ioC \bullet \\
\qquad\qquad\qquad\quad \langle inI \rangle \frown is \frown \langle inF \rangle \mapsto (c, \langle last\ (is \frown \langle inF \rangle) \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad startBatch\ ioC \fatsemi runBatch^{\top}\ ioC\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, last\ (is \frown \langle inF \rangle)) \notin \mathrm{dom}\ ioFinal\ ioC \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists a : \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad\qquad \langle inI \rangle \frown is \frown \langle inF \rangle \mapsto (a, \langle last\ (is \frown \langle inF \rangle) \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad\quad startBatch\ ioA \fatsemi runBatch^{\top}\ ioA\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, last\ (is \frown \langle inF \rangle)) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad [\text{property of } \# \text{ and } \frown] \\
\qquad startBatch\ ioC \mathbin{;} runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioC \\
\qquad\qquad \subseteq \\
\qquad\qquad startBatch\ ioA \mathbin{;} runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioA \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \mathbin{;} runBatch^\top\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch^\top\ ioC \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{ outI : Value;\ outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad \#outs' < n \bullet \langle outI \rangle \frown outs' \} \\
\qquad\qquad \subseteq \\
\qquad\qquad startBatch\ ioA \mathbin{;} runBatch^\top\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch^\top\ ioA \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{ outI : Value;\ outs' : \mathrm{seq}\ Value \mid \\
\qquad\qquad\qquad\qquad\quad \#outs' < n \bullet \langle outI \rangle \frown outs' \} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ inI, outI, inF : Value;\ is, outs' : \mathrm{seq}\ Value \mid \\
\qquad \#is = n\ \wedge\ \#outs' = n \bullet \\
\qquad\qquad \exists\, c : ioState\ ioC \bullet \\
\qquad\qquad\qquad\quad \langle inI \rangle \frown is \frown \langle inF \rangle \mapsto (c, \langle last\,(is \frown \langle inF \rangle) \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad startBatch\ ioC \mathbin{;} runBatch^\top\ ioC\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, last\,(is \frown \langle inF \rangle)) \notin \mathrm{dom}\ ioFinal\ ioC \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad\qquad \langle inI \rangle \frown is \frown \langle inF \rangle \mapsto (a, \langle last\,(is \frown \langle inF \rangle) \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad\quad startBatch\ ioA \mathbin{;} runBatch^\top\ ioA\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, last\,(is \frown \langle inF \rangle)) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\mathrm{def}^{\,n} \text{ of } \mathit{last}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\quad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{is}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{is} = n\ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[re-naming: ins for is]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioC} \\
\qquad\qquad\qquad \rhd \\
\qquad\qquad\qquad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} \mathit{stopBatch}^\top\ \mathit{ioA} \\
\qquad\qquad\qquad\quad \rhd \\
\qquad\qquad\qquad\quad \{\mathit{outI} : \mathit{Value};\ \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad\qquad\qquad\qquad\quad \#\mathit{outs}' < n \bullet \langle \mathit{outI} \rangle \frown \mathit{outs}'\} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill [\mathrm{def}^{\,n} \text{ of } \subseteq] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \# \mathit{ins} = n + 1\ \wedge\ \# \mathit{outs}' < n \bullet \\
\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathbin{;} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \# \mathit{ins} = n\ \wedge\ \# \mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\qquad\quad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill \text{[one point rule]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n, m : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad m < n \\
\qquad \wedge \\
\qquad \#\mathit{ins} = n + 1 \ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbin{⨾} \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathbin{⨾} \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n \ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad\qquad\qquad\qquad\qquad [\text{substitution } \mathit{is} \frown \mathit{is}' \text{ for } \mathit{ins}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\quad \subseteq \\
\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n, m : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{is}, \mathit{is}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad m < n \\
\qquad \wedge \\
\qquad \#\mathit{is} = m \ \wedge\ \#\mathit{is}' = (n - m + 1)\ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \mathit{is}' \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \mathit{is}' \mapsto \langle \mathit{outI} \rangle \frown \mathit{outs}' \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \fatsemi \mathit{stopBatch}^{\top}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n \ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\quad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (c, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a : \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad [\text{def}^n\text{s of } \fatsemi,\ \mathit{runBatch}^\top \text{ and } \mathit{stopBatch}^\top] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n, m : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{is}, \mathit{is}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad m < n \\
\qquad \wedge \\
\qquad \#\mathit{is} = m\ \wedge\ \#\mathit{is}' = (n - m + 1)\ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \mathit{is}' \mapsto (c, \mathit{is}', \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^m) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (c, \mathit{head}\ \mathit{is}') \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathit{ioState}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{is} \frown \mathit{is}' \mapsto (c, \mathit{is}', \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^m) \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (c, \mathit{head}\ \mathit{is}') \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^\top\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (c, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^\top\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad [\mathrm{def}^n \text{ of } \mathit{runBatch} \text{ etc}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n, m : \mathbb{N};\ \mathit{inI}, \mathit{outI} : \mathit{Value};\ \mathit{is}, \mathit{is}', \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad m < n \\
\qquad \wedge \\
\qquad \#\mathit{is} = m\ \wedge\ \#\mathit{is}' = (n - m + 1)\ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\qquad \exists\, \mathit{cs} : \operatorname{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = m + 1 \bullet \\
\qquad\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad \forall\, i : 1 \mathinner{..} m \bullet \\
\qquad\qquad\qquad\qquad\qquad (\mathit{cs}\ i, (\mathit{is} \frown \mathit{is}')\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (\mathit{cs}\ (m+1), \mathit{head}\ \mathit{is}') \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, \mathit{as} : \operatorname{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = m + 1 \bullet \\
\qquad\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad \forall\, i : 1 \mathinner{..} m \bullet \\
\qquad\qquad\qquad\qquad\qquad (\mathit{as}\ i, (\mathit{is} \frown \mathit{is}')\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (\mathit{as}\ (m+1), \mathit{head}\ \mathit{is}') \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (c, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad [\text{substitution: } \mathit{head}\ \mathit{is}' = \mathit{in}] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\quad \subseteq \\
\qquad\quad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \fatsemi \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, n, m : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{is}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad m < n \\
\qquad \wedge \\
\qquad \#\mathit{is} = m \ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\quad \exists\, \mathit{cs} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = m + 1 \bullet \\
\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \forall\, i : 1 \,..\, m \bullet \\
\qquad\qquad\qquad\quad (\mathit{cs}\ i, \mathit{is}\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (\mathit{cs}\ (m+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, \mathit{as} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = m + 1 \bullet \\
\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \forall\, i : 1 \,..\, m \bullet \\
\qquad\qquad\qquad\quad (\mathit{as}\ i, \mathit{is}\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (\mathit{as}\ (m+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n \ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\quad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioC} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \fatsemi \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill \text{[property of } \forall] \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\quad \subseteq \\
\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \wedge \\
\quad \forall\, m : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{is}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{is} = m\ \wedge\ \#\mathit{outs}' = m \bullet \\
\qquad\quad \exists\, \mathit{cs} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = m + 1 \bullet \\
\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} m \bullet \\
\qquad\qquad\qquad (\mathit{cs}\ i, \mathit{is}\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\quad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (\mathit{cs}\ (m+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, \mathit{as} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = m + 1 \bullet \\
\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} m \bullet \\
\qquad\qquad\qquad (\mathit{as}\ i, \mathit{is}\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\quad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (\mathit{as}\ (m+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \wedge\ \#\mathit{outs}' = n \bullet \\
\qquad\quad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill \text{[re-naming]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, \mathit{cs} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = n + 1 \bullet \\
\qquad\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad \forall\, i : 1 \,..\, n \bullet \\
\qquad\qquad\qquad\qquad (\mathit{cs}\ i, \mathit{ins}\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad (\mathit{cs}\ (n+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, \mathit{as} : \mathrm{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = n + 1 \bullet \\
\qquad\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad \forall\, i : 1 \,..\, n \bullet \\
\qquad\qquad\qquad\qquad (\mathit{as}\ i, \mathit{ins}\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad (\mathit{as}\ (n+1), \mathit{in}) \notin \mathrm{dom}\ \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \mathrm{seq}\ \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\quad \land \\
\qquad\qquad\qquad\quad (c, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \mathrm{dom}\ \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \forall\, n : \mathbb{N} \bullet \hfill \text{[property of } \land\text{]} \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioC} \\
\qquad\qquad \subseteq \\
\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \mathbin{⨾} \mathit{stopBatch}\ \mathit{ioA} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, \mathit{cs} : \operatorname{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = n + 1 \bullet \\
\qquad\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad\qquad\qquad (\mathit{cs}\ i, \mathit{ins}\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (\mathit{cs}\ (n+1), \mathit{in}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, \mathit{as} : \operatorname{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = n + 1 \bullet \\
\qquad\qquad\qquad\qquad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad\qquad\qquad (\mathit{as}\ i, \mathit{ins}\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) \\
\qquad\qquad\qquad\qquad\qquad\qquad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (\mathit{as}\ (n+1), \mathit{in}) \notin \operatorname{dom} \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} \\
\quad \land \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \operatorname{seq} \mathit{Value} \mid \\
\qquad \#\mathit{ins} = n\ \land\ \#\mathit{outs}' = n \bullet \\
\qquad\qquad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet \\
\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad (c, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioC} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA} \bullet \\
\qquad\qquad\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioA} \mathbin{⨾} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \mathrm{op} \rangle^{n}) \\
\qquad\qquad\qquad\qquad\qquad \land \\
\qquad\qquad\qquad\qquad\qquad (a, \mathit{inF}) \notin \operatorname{dom} \mathit{ioFinal}\ \mathit{ioA}
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \text{true} \qquad \text{[property of } \wedge \text{]} \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \mathbin{;} runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioC \\
\qquad\quad \subseteq \\
\qquad\quad startBatch\ ioA \mathbin{;} runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioA \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \mathbin{;} runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioC \\
\qquad\quad \subseteq \\
\qquad\quad startBatch\ ioA \mathbin{;} runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{;} stopBatch\ ioA \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ inI, outI, in : Value;\ ins, outs' : \mathrm{seq}\ Value \mid \\
\qquad \#ins = n\ \wedge\ \#outs' = n \bullet \\
\qquad\quad \exists\, cs : \mathrm{seq}(ioState\ ioC) \mid \#cs = n + 1 \bullet \\
\qquad\qquad\quad inI \mapsto (cs\ 1, outI) \in ioInit\ ioC \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad (cs\ i, ins\ i) \mapsto (cs\ (i+1), outs'\ i) \\
\qquad\qquad\qquad\quad \in ioOps\ ioC\ op \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (cs\ (n+1), in) \notin \mathrm{dom}\ ioOps\ ioC\ op \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, as : \mathrm{seq}(ioState\ ioA) \mid \#as = n + 1 \bullet \\
\qquad\qquad\quad inI \mapsto (as\ 1, outI) \in ioInit\ ioA \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad (as\ i, ins\ i) \mapsto (as\ (i+1), outs'\ i) \\
\qquad\qquad\qquad\quad \in ioOps\ ioA\ op \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (as\ (n+1), in) \notin \mathrm{dom}\ ioOps\ ioA\ op \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ inI, outI, inF : Value;\ ins, outs' : \mathrm{seq}\ Value \mid \\
\qquad \#ins = n\ \wedge\ \#outs' = n \bullet \\
\qquad\quad \exists\, c : ioState\ ioC \bullet \\
\qquad\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad startBatch\ ioC \mathbin{;} runBatch^{\top}\ ioC\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, inF) \notin \mathrm{dom}\ ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a : \mathrm{dom}\ ioFinal\ ioA \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\quad startBatch\ ioA \mathbin{;} runBatch^{\top}\ ioA\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, inF) \notin \mathrm{dom}\ ioFinal\ ioA
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow \text{true} & [\text{associativity of } \wedge] \\
\quad \wedge & \\
\quad (\forall\, n : \mathbb{N} \bullet & \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \text{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad\quad \subseteq & \\
\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \text{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA}) & \\
\quad \wedge & \\
\quad (\forall\, n : \mathbb{N} \bullet & \\
\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}\ \mathit{ioC}\ (\langle \text{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioC} & \\
\qquad\quad \subseteq & \\
\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}\ \mathit{ioA}\ (\langle \text{op} \rangle^n) \mathbin{;} \mathit{stopBatch}\ \mathit{ioA} & \\
\quad \wedge & \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{in} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \text{seq}\ \mathit{Value} \mid & \\
\qquad \#\mathit{ins} = n\ \wedge\ \#\mathit{outs}' = n \bullet & \\
\qquad\quad \exists\, \mathit{cs} : \text{seq}(\mathit{ioState}\ \mathit{ioC}) \mid \#\mathit{cs} = n + 1 \bullet & \\
\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{cs}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioC} & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet & \\
\qquad\qquad\qquad (\mathit{cs}\ i, \mathit{ins}\ i) \mapsto (\mathit{cs}\ (i+1), \mathit{outs}'\ i) & \\
\qquad\qquad\qquad\quad \in \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad (\mathit{cs}\ (n+1), \mathit{in}) \notin \text{dom}\ \mathit{ioOps}\ \mathit{ioC}\ \mathit{op} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \exists\, \mathit{as} : \text{seq}(\mathit{ioState}\ \mathit{ioA}) \mid \#\mathit{as} = n + 1 \bullet & \\
\qquad\qquad\quad \mathit{inI} \mapsto (\mathit{as}\ 1, \mathit{outI}) \in \mathit{ioInit}\ \mathit{ioA} & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet & \\
\qquad\qquad\qquad (\mathit{as}\ i, \mathit{ins}\ i) \mapsto (\mathit{as}\ (i+1), \mathit{outs}'\ i) & \\
\qquad\qquad\qquad\quad \in \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad (\mathit{as}\ (n+1), \mathit{in}) \notin \text{dom}\ \mathit{ioOps}\ \mathit{ioA}\ \mathit{op} & \\
\quad \wedge & \\
\quad \forall\, n : \mathbb{N};\ \mathit{inI}, \mathit{outI}, \mathit{inF} : \mathit{Value};\ \mathit{ins}, \mathit{outs}' : \text{seq}\ \mathit{Value} \mid & \\
\qquad \#\mathit{ins} = n\ \wedge\ \#\mathit{outs}' = n \bullet & \\
\qquad\quad \exists\, c : \mathit{ioState}\ \mathit{ioC} \bullet & \\
\qquad\qquad\quad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (c, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in & \\
\qquad\qquad\qquad \mathit{startBatch}\ \mathit{ioC} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioC}\ (\langle \text{op} \rangle^n) & \\
\qquad\qquad\quad \wedge & \\
\qquad\qquad\quad (c, \mathit{inF}) \notin \text{dom}\ \mathit{ioFinal}\ \mathit{ioC} & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \exists\, a : \text{dom}\ \mathit{ioFinal}\ \mathit{ioA} \bullet & \\
\qquad\qquad\qquad \langle \mathit{inI} \rangle \frown \mathit{ins} \frown \langle \mathit{inF} \rangle \mapsto (a, \langle \mathit{inF} \rangle, \langle \mathit{outI} \rangle \frown \mathit{outs}') \in & \\
\qquad\qquad\qquad\quad \mathit{startBatch}\ \mathit{ioA} \mathbin{;} \mathit{runBatch}^{\top}\ \mathit{ioA}\ (\langle \text{op} \rangle^n) & \\
\qquad\qquad\qquad \wedge & \\
\qquad\qquad\qquad (a, \mathit{inF}) \notin \text{dom}\ \mathit{ioFinal}\ \mathit{ioA}) &
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\{X : \mathbb{P}\,\Sigma \mid \hfill [\text{Lemma 9}] \\
\qquad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\ ioC \bullet \\
\qquad\quad \mathrm{init}.inI.outI \in X \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, outI' : \operatorname{ran}(ioInit\ ioC (\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\quad \bullet (\ \langle\rangle\ , X)\} \\
\qquad \subseteq \\
\qquad \{X : \mathbb{P}\,\Sigma \mid \\
\qquad\quad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\ ioA \bullet \\
\qquad\qquad \mathrm{init}.inI.outI \in X \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists\, outI' : \operatorname{ran}(ioInit\ ioA (\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad \bullet (\ \langle\rangle\ , X)\}\ ) \\
\ \wedge \\
\ (\forall\, n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \mathbin{⨾} runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} stopBatch\ ioC \\
\qquad\quad \subseteq \\
\qquad\quad startBatch\ ioA \mathbin{⨾} runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} stopBatch\ ioA) \\
\ \wedge \\
\ (\forall\, n : \mathbb{N} \bullet \\
\qquad startBatch\ ioC \mathbin{⨾} runBatch\ ioC\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} stopBatch\ ioC \\
\qquad\quad \subseteq \\
\qquad\quad startBatch\ ioA \mathbin{⨾} runBatch\ ioA\ (\langle \mathrm{op} \rangle^n) \mathbin{⨾} stopBatch\ ioA \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ inI, outI, in : Value;\ ins, outs' : \operatorname{seq} Value \mid \\
\qquad \#ins = n\ \wedge\ \#outs' = n \bullet \\
\qquad\quad \exists\, cs : \operatorname{seq}(ioState\ ioC) \mid \#cs = n + 1 \bullet \\
\qquad\qquad\quad inI \mapsto (cs\ 1, outI) \in ioInit\ ioC \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \ldots n \bullet \\
\qquad\qquad\qquad (cs\ i, ins\ i) \mapsto (cs\ (i+1), outs'\ i) \\
\qquad\qquad\qquad\quad \in ioOps\ ioC\ op \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (cs\ (n+1), in) \notin \operatorname{dom} ioOps\ ioC\ op \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, as : \operatorname{seq}(ioState\ ioA) \mid \#as = n + 1 \bullet \\
\qquad\qquad\quad inI \mapsto (as\ 1, outI) \in ioInit\ ioA \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, i : 1 \ldots n \bullet \\
\qquad\qquad\qquad (as\ i, ins\ i) \mapsto (as\ (i+1), outs'\ i) \\
\qquad\qquad\qquad\quad \in ioOps\ ioA\ op \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (as\ (n+1), in) \notin \operatorname{dom} ioOps\ ioA\ op \\
\quad \wedge \\
\quad \forall\, n : \mathbb{N};\ inI, outI, inF : Value;\ ins, outs' : \operatorname{seq} Value \mid \\
\qquad \#ins = n\ \wedge\ \#outs' = n \bullet \\
\qquad\quad \exists\, c : ioState\ ioC \bullet \\
\qquad\qquad\quad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (c, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad startBatch\ ioC \mathbin{⨾} runBatch^{\top}\ ioC\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad (c, inF) \notin \operatorname{dom} ioFinal\ ioC \\
\qquad\qquad \Rightarrow \\
\qquad\qquad \exists\, a : \operatorname{dom} ioFinal\ ioA \bullet \\
\qquad\qquad\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto (a, \langle inF \rangle, \langle outI \rangle \frown outs') \in \\
\qquad\qquad\qquad\quad startBatch\ ioA \mathbin{⨾} runBatch^{\top}\ ioA\ (\langle \mathrm{op} \rangle^n) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (a, inF) \notin \operatorname{dom} ioFinal\ ioA)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow (\{X : \mathbb{P}\,\Sigma \mid \hfill [\text{Lemma 10}] \\
\qquad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\; ioC \bullet \\
\qquad\qquad \mathrm{init}.inI.outI \in X \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, outI' : \mathrm{ran}(ioInit\; ioC \llparenthesis \{inI\} \rrparenthesis) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad \bullet (\langle\rangle, X)\} \\
\qquad \subseteq \\
\qquad \{X : \mathbb{P}\,\Sigma \mid \\
\qquad\qquad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\; ioA \bullet \\
\qquad\qquad\qquad \mathrm{init}.inI.outI \in X \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad \exists\, outI' : \mathrm{ran}(ioInit\; ioA \llparenthesis \{inI\} \rrparenthesis) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad\qquad \bullet (\langle\rangle, X)\}) \\
\quad \wedge \\
\quad (\{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value;\; ins, outs : \mathrm{seq}\, Value \mid \\
\qquad\quad \#ins = \#outs = n \\
\qquad\quad \wedge \\
\qquad\quad \langle inI\rangle \frown ins \frown \langle inF\rangle \mapsto \langle outI\rangle \frown outs \frown \langle outF\rangle \in \\
\qquad\qquad\qquad startBatch\; ioC \fatsemi runBatch\; ioC \langle op\rangle^n \fatsemi stopBatch\; ioC \\
\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI\rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \frown \langle \mathrm{final}.inF.outF\rangle, X)\} \\
\qquad \subseteq \\
\qquad \{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value;\; ins, outs : \mathrm{seq}\, Value \mid \\
\qquad\qquad \#ins = \#outs = n \\
\qquad\qquad \wedge \\
\qquad\qquad \langle inI\rangle \frown ins \frown \langle inF\rangle \mapsto \langle outI\rangle \frown outs \frown \langle outF\rangle \in \\
\qquad\qquad\qquad\qquad startBatch\; ioA \fatsemi runBatch\; ioA \langle op\rangle^n \fatsemi stopBatch\; ioA \\
\qquad\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI\rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r)\rangle \frown \langle \mathrm{final}.inF.outF\rangle, X)\}) \\
\quad \wedge \\
\quad (\forall\, n : \mathbb{N} \bullet \\
\qquad\quad startBatch\; ioC \fatsemi runBatch\; ioC\, (\langle \mathrm{op}\rangle^n) \fatsemi stopBatch\; ioC \\
\qquad\qquad \subseteq \\
\qquad\qquad startBatch\; ioA \fatsemi runBatch\; ioA\, (\langle \mathrm{op}\rangle^n) \fatsemi stopBatch\; ioA \\
\quad\; \wedge \\
\quad\; \forall\, n : \mathbb{N};\; inI, outI, in : Value;\; ins, outs' : \mathrm{seq}\, Value \mid \\
\qquad\quad \#ins = n \;\wedge\; \#outs' = n \bullet \\
\qquad\qquad \exists\, cs : \mathrm{seq}(ioState\; ioC) \mid \#cs = n + 1 \bullet \\
\qquad\qquad\qquad\quad inI \mapsto (cs\, 1, outI) \in ioInit\; ioC \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad\qquad (cs\; i, ins\; i) \mapsto (cs\,(i+1), outs'\; i) \\
\qquad\qquad\qquad\qquad\qquad \in ioOps\; ioC\; op \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (cs\,(n+1), in) \notin \mathrm{dom}\, ioOps\; ioC\; op \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, as : \mathrm{seq}(ioState\; ioA) \mid \#as = n + 1 \bullet \\
\qquad\qquad\qquad\quad inI \mapsto (as\, 1, outI) \in ioInit\; ioA \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad \forall\, i : 1 \mathinner{..} n \bullet \\
\qquad\qquad\qquad\qquad (as\; i, ins\; i) \mapsto (as\,(i+1), outs'\; i) \\
\qquad\qquad\qquad\qquad\qquad \in ioOps\; ioA\; op \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (as\,(n+1), in) \notin \mathrm{dom}\, ioOps\; ioA\; op \\
\quad\; \wedge \\
\quad\; \forall\, n : \mathbb{N};\; inI, outI, inF : Value;\; ins, outs' : \mathrm{seq}\, Value \mid \\
\qquad\quad \#ins = n \;\wedge\; \#outs' = n \bullet \\
\qquad\qquad \exists\, c : ioState\; ioC \bullet \\
\qquad\qquad\qquad\quad \langle inI\rangle \frown ins \frown \langle inF\rangle \mapsto (c, \langle inF\rangle, \langle outI\rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad startBatch\; ioC \fatsemi runBatch^{\top}\; ioC\, (\langle \mathrm{op}\rangle^n) \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad (c, inF) \notin \mathrm{dom}\, ioFinal\; ioC \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, a : \mathrm{dom}\, ioFinal\; ioA \bullet \\
\qquad\qquad\qquad\qquad \langle inI\rangle \frown ins \frown \langle inF\rangle \mapsto (a, \langle inF\rangle, \langle outI\rangle \frown outs') \in \\
\qquad\qquad\qquad\qquad\qquad startBatch\; ioA \fatsemi runBatch^{\top}\; ioA\, (\langle \mathrm{op}\rangle^n) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad (a, inF) \notin \mathrm{dom}\, ioFinal\; ioA)
\end{array}
$$

$$
\begin{array}{l}
\Leftrightarrow \{X : \mathbb{P}\,\Sigma \mid \hfill \text{[Lemma 11]} \\
\qquad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\; ioC \bullet \\
\qquad\qquad \mathrm{init}.inI.outI \in X \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists\, outI' : \operatorname{ran}(ioInit\; ioC(\!|\, \{inI\} \,|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad \bullet\, (\,\langle\rangle\,, X)\} \\
\qquad \subseteq \\
\qquad \{X : \mathbb{P}\,\Sigma \mid \\
\qquad\qquad \forall\, inI, outI : Value \mid inI \in \operatorname{dom} ioInit\; ioA \bullet \\
\qquad\qquad\qquad \mathrm{init}.inI.outI \in X \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad \exists\, outI' : \operatorname{ran}(ioInit\; ioA(\!|\, \{inI\} \,|\!)) \bullet \mathrm{init}.inI.outI' \notin X \\
\qquad\qquad\qquad\qquad\qquad \bullet\, (\,\langle\rangle\,, X)\} \\
\wedge \\
\{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value;\; ins, outs : \operatorname{seq} Value \mid \\
\qquad \#ins = \#outs = n \\
\qquad \wedge \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad\qquad startBatch\; ioC \mathbin{⨟} runBatch\; ioC \langle op \rangle^{n} \mathbin{⨟} stopBatch\; ioC \\
\qquad\qquad \bullet\, (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} \\
\qquad \subseteq \\
\qquad \{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI, inF, outF : Value;\; ins, outs : \operatorname{seq} Value \mid \\
\qquad\qquad \#ins = \#outs = n \\
\qquad\qquad \wedge \\
\qquad\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in \\
\qquad\qquad\qquad\qquad startBatch\; ioA \mathbin{⨟} runBatch\; ioA \langle op \rangle^{n} \mathbin{⨟} stopBatch\; ioA \\
\qquad\qquad\qquad \bullet\, (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} \\
\wedge \\
\{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \operatorname{seq} Value \mid \\
\qquad (\, \#ins = \#outs = n \\
\qquad\;\; \wedge \\
\qquad\;\; \exists\, c : Local \bullet \\
\qquad\qquad\quad \langle inI \rangle \frown ins \mapsto (c,\, \langle\rangle\,, \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad startBatch\; ioC \mathbin{⨟} runBatch\; ioC \langle op \rangle^{n} \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\qquad out \notin \operatorname{ran}(ioOps\; ioC\; op(\!|\, \{(c, in)\} \,|\!)) \\
\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad \exists\, out' : \operatorname{ran}(ioOps\; ioC\; op(\!|\, \{(c, in)\} \,|\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\qquad\quad \wedge \\
\qquad\qquad\quad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\qquad outF \notin \operatorname{ran}(ioFinal\; ioC(\!|\, \{(c, inF)\} \,|\!)) \\
\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad \exists\, outF' : \operatorname{ran}(ioOps\; ioC\; op(\!|\, \{(c, in)\} \,|\!)) \bullet \mathrm{final}.inF.outF' \notin X\,) \\
\qquad\qquad\qquad\quad \bullet\, (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle, X)\} \\
\qquad \subseteq \\
\qquad \{n : \mathbb{N};\; X : \mathbb{P}\,\Sigma;\; inI, outI : Value;\; ins, outs : \operatorname{seq} Value \mid \\
\qquad\qquad (\, \#ins = \#outs = n \\
\qquad\qquad\;\; \wedge \\
\qquad\qquad\;\; \exists\, a : Local \bullet \\
\qquad\qquad\qquad\quad \langle inI \rangle \frown ins \mapsto (a,\, \langle\rangle\,, \langle outI \rangle \frown outs) \in \\
\qquad\qquad\qquad\qquad startBatch\; ioA \mathbin{⨟} runBatch\; ioA \langle op \rangle^{n} \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow \\
\qquad\qquad\qquad\qquad out \notin \operatorname{ran}(ioOps\; ioA\; op(\!|\, \{(a, in)\} \,|\!)) \\
\qquad\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad\qquad \exists\, out' : \operatorname{ran}(ioOps\; ioA\; op(\!|\, \{(a, in)\} \,|\!)) \bullet \mathrm{op}.in.out' \notin X \\
\qquad\qquad\qquad\quad \wedge \\
\qquad\qquad\qquad\quad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow \\
\qquad\qquad\qquad\qquad outF \notin \operatorname{ran}(ioFinal\; ioA(\!|\, \{(a, inF)\} \,|\!)) \\
\qquad\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad\qquad \exists\, outF' : \operatorname{ran}(ioOps\; ioA\; op(\!|\, \{(a, in)\} \,|\!)) \bullet \mathrm{final}.inF.outF' \notin X\,) \\
\qquad\qquad\qquad\qquad\quad \bullet\, (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{..} n \bullet \mathrm{op}.(ins\; r).(outs\; r) \rangle, X)\}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow (\{X : \mathbb{P}\,\Sigma \mid & \text{[set theory]} \\
\quad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\ ioC \bullet & \\
\qquad \mathrm{init}.inI.outI \in X & \\
\qquad\quad \Rightarrow & \\
\qquad\quad \exists\, outI' : \mathrm{ran}(ioInit\ ioC (\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X & \\
\qquad\qquad \bullet\ (\ \langle\rangle\ , X)\} & \\
\ \cup & \\
\ \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid & \\
\qquad \#ins = \#outs = n & \\
\qquad \wedge & \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in & \\
\qquad\qquad startBatch\ ioC \,⨾\, runBatch\ ioC \langle op \rangle^{n} \,⨾\, stopBatch\ ioC & \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} & \\
\ \cup & \\
\ \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid & \\
\qquad (\ \#ins = \#outs = n & \\
\qquad\ \ \wedge & \\
\qquad\ \ \exists\, c : Local \bullet & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (c,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in & \\
\qquad\qquad\quad startBatch\ ioC \,⨾\, runBatch\ ioC \langle op \rangle^{n} & \\
\qquad\qquad \wedge & \\
\qquad\qquad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow & \\
\qquad\qquad\quad out \notin \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) & \\
\qquad\qquad\quad \vee & \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X & \\
\qquad\qquad \wedge & \\
\qquad\qquad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow & \\
\qquad\qquad\quad outF \notin \mathrm{ran}(ioFinal\ ioC (\!|\ \{(c, inF)\}\ |\!)) & \\
\qquad\qquad\quad \vee & \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioC\ op (\!|\ \{(c, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) & \\
\qquad\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}) & \\
\quad \subseteq & \\
\quad (\{X : \mathbb{P}\,\Sigma \mid & \\
\qquad \forall\, inI, outI : Value \mid inI \in \mathrm{dom}\, ioInit\ ioA \bullet & \\
\qquad\quad \mathrm{init}.inI.outI \in X & \\
\qquad\qquad \Rightarrow & \\
\qquad\qquad \exists\, outI' : \mathrm{ran}(ioInit\ ioA (\!|\ \{inI\}\ |\!)) \bullet \mathrm{init}.inI.outI' \notin X & \\
\qquad\qquad\quad \bullet\ (\ \langle\rangle\ , X)\} & \\
\quad \cup & \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI, inF, outF : Value;\ ins, outs : \mathrm{seq}\ Value \mid & \\
\qquad \#ins = \#outs = n & \\
\qquad \wedge & \\
\qquad \langle inI \rangle \frown ins \frown \langle inF \rangle \mapsto \langle outI \rangle \frown outs \frown \langle outF \rangle \in & \\
\qquad\qquad startBatch\ ioA \,⨾\, runBatch\ ioA \langle op \rangle^{n} \,⨾\, stopBatch\ ioA & \\
\qquad\quad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle \frown \langle \mathrm{final}.inF.outF \rangle, X)\} & \\
\quad \cup & \\
\quad \{n : \mathbb{N};\ X : \mathbb{P}\,\Sigma;\ inI, outI : Value;\ ins, outs : \mathrm{seq}\ Value \mid & \\
\qquad (\ \#ins = \#outs = n & \\
\qquad\ \ \wedge & \\
\qquad\ \ \exists\, a : Local \bullet & \\
\qquad\qquad \langle inI \rangle \frown ins \mapsto (a,\ \langle\rangle\ , \langle outI \rangle \frown outs) \in & \\
\qquad\qquad\quad startBatch\ ioA \,⨾\, runBatch\ ioA \langle op \rangle^{n} & \\
\qquad\qquad \wedge & \\
\qquad\qquad \forall\, in, out : Value \bullet \mathrm{op}.in.out \in X \Rightarrow & \\
\qquad\qquad\quad out \notin \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) & \\
\qquad\qquad\quad \vee & \\
\qquad\qquad\quad \exists\, out' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{op}.in.out' \notin X & \\
\qquad\qquad \wedge & \\
\qquad\qquad \forall\, inF, outF : Value \bullet \mathrm{final}.inF.outF \in X \Rightarrow & \\
\qquad\qquad\quad outF \notin \mathrm{ran}(ioFinal\ ioA (\!|\ \{(a, inF)\}\ |\!)) & \\
\qquad\qquad\quad \vee & \\
\qquad\qquad\quad \exists\, outF' : \mathrm{ran}(ioOps\ ioA\ op (\!|\ \{(a, in)\}\ |\!)) \bullet \mathrm{final}.inF.outF' \notin X\ ) & \\
\qquad\qquad\qquad \bullet (\langle \mathrm{init}.inI.outI \rangle \frown \langle r : 1 \mathinner{\ldotp\ldotp} n \bullet \mathrm{op}.(ins\ r).(outs\ r) \rangle, X)\}) &
\end{array}
$$

$$\Leftrightarrow \mathsf{ioProcess}\,(ioA) \;\sqsubseteq_F\; \mathsf{ioProcess}\,(ioC) \qquad [\text{def}^{\,n} \text{ of } \sqsubseteq_F]$$

We have proved the desired result.

Bibliography

[BDW99] C. Bolton, J. Davies, and J. Woodcock. On the refinement and simulation of data types and processes. In K. Araki, A. Galloway, and K. Taguchi, editors, *International conference on Integrated Formal Methods 1999 (IFM'99)*, pages 273–292. Springer, July 1999.

[Bol98] C. Bolton. On the refinement of abstract data types and their process equivalents. Master's thesis, Oxford University Computing Laboratory, Oxford University, 1998.

[For95] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement: FDR2*, December 1995. Preliminary Manual.

[HHS87] C. A. R. Hoare, J. He, and J. W. Sanders. Prespecification in data refinement. *Information Processing Letters*, 25(2):71 – 76, 1987.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[Ros98] A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall Series in Computer Science, 1998.

[Spi92] J.M. Spivey. *The Z notation: a reference manual.* Prentice-Hall International, March 1992.

[WD96] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Proof and Refinement.* Prentice Hall International Series in Computer Science, 1996.