Process-Local Static Analysis of Synchronous Processes

Midtgaard, Jan; Nielson, Flemming; Nielson, Hanne Riis

Published in:
Static Analysis

Link to article, DOI:
10.1007/978-3-319-99725-4_18

Publication date:
2018

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Midtgaard, J., Nielson, F., & Nielson, H. R. (2018). Process-Local Static Analysis of Synchronous Processes. In *Static Analysis* (pp. 284-305). Springer. Lecture Notes in Computer Science, Vol.. 11002
https://doi.org/10.1007/978-3-319-99725-4_18

This is the authors' full version including proofs.
The original version appeared in the 25th Static Analysis Symposium, 2018 (SAS'18)

Process-Local Static Analysis of Synchronous Processes

Jan Midtgaard[1], Flemming Nielson[2], and Hanne Riis Nielson[2]

[1] The Maersk Mc-Kinney Moller Institute, University of Southern Denmark
[2] DTU Compute, Technical University of Denmark

Abstract. We develop a modular approach to statically analyse imperative processes communicating by synchronous message passing. The approach is modular in that it only needs to analyze one process at a time, but will in general have to do so repeatedly. The approach combines lattice-valued regular expressions to capture network communication with a dedicated shuffle operator for composing individual process analysis results. We present both a soundness proof and a prototype implementation of the approach for a synchronous subset of the Go programming language. Overall our approach tackles the combinatorial explosion of concurrent programs by suitable static analysis approximations, thereby lifting traditional sequential analysis techniques to a concurrent setting.

1 Introduction

Concurrent software surrounds us: whether as an *app* on a mobile phone communicating with a server, in the software business where a system has been structured as a service-oriented architecture, or at the data center where processes spread on many processors to collectively solve a computational query, they are all structured as software processes communicating by some form of message passing. The past decades contain a line of work towards ensuring correctness of such software: The model checking community has developed techniques for validating such distributive designs and the types community has developed *session types* for checking the overall communication structure. Within the static analysis community a line of work has pursued static analysis of process calculi (which may themselves be viewed as suitable process abstractions).

In this work we develop a static analysis approach that works directly at the source code level and addresses how safety properties of a distributed program may depend on intricate details involving both the order and content of the network communication. Rather than risk a combinatorial explosion by computing a collective state of all involved processes, our approach captures the network communication between a number of synchronous, message-passing processes with a dedicated abstract domain. This approach allows us to analyze each process separately. We then combine the analysis results of individual processes with a dedicated shuffle operator for the domain. We prove soundness of the analysis with respect to an operational semantics for a subset of Go and discuss a prototype implementation of the approach.

Consider the Go program in Fig. 1. It declares two common channels `ch1` and `ch2`, spawns off two processes (*go-routines*), and proceeds to the main read-statement at the bottom. The first process in line 6 attempts to send 1 on channel `ch1` and 2 on channel `ch2`. The second process in line 7 reads a value (1) from channel `ch1` into variable `x` and sends the value of `x+1` (2) on channel `ch2`. Finally the read statement

```
1   package main
2
3   func main() {
4           ch1 := make(chan int);
5           ch2 := make(chan int);
6           go func() { ch1 <-1; ch2 <-2; }()
7           go func() { var x int;
8                       x = <-ch1; ch2 <- x+1; }()
9           var y int;
10          y = <- ch2;
11  }
```

Fig. 1: An example Go program

in line 10 reads a value from `ch2`. Under worst-case intra-process analysis assumptions this read could receive any value and bind it to `y`. This is also the result of a first iteration of our intra-process analysis. From this first intra-process analysis result we can read off that the three processes perform (the prefix-closure of) the network actions $\texttt{ch1}![1;1] \cdot \texttt{ch2}![2;2]$, $\texttt{ch1}?[-\infty;+\infty] \cdot \texttt{ch2}![-\infty;+\infty]$, and $\texttt{ch2}?[-\infty;+\infty]$ respectively, here expressed as *lattice-valued regular expressions* with channel-tagged intervals. By *shuffling* the first and third result and performing intra-process reanalysis of the second process under this stronger assumption, we learn that it actually performs (the prefix-closure of) the network actions $\texttt{ch1}?[1;1] \cdot \texttt{ch2}![2;2]$. Finally we shuffle this result with the result from the first process and run a third round of intra-process reanalysis to learn that the value read from `ch2` and assigned to `y` is constant $[2;2]$.

2 Language

We consider an imperative core language extended with primitives for synchronous message passing between individual processes, as illustrated by the above example. The core language is designed to be a genuine subset of Go (restricted to synchronous message passing), which we term *nano-Go*. Because of our restrictions, programs in nano-Go consist of a fixed number of top-level processes communicating through a fixed number of channels:

```
func main() {
        ch_1 := make(chan int) ... ch_k := make(chan int)
        go func() { s_1   }()
        ...
        go func() { s_{n-1} }()
        s_n
}
```

As such, the programs spawn off n processes and can thereby conveniently be described by their process bodies $s_1, \ldots, s_n$ from an abstract syntax point of view. We provide a BNF grammar of the process language in Fig. 2. Each process is defined by a composite statement (ending in a blocking `select` $\{\ \}$ statement) and has access to a process-local environment of pre-declared variables.

The statements of the language are mostly self-explanatory. `select` $\{ a_1 \ldots a_n \}$ non-deterministically chooses between a list of read and write cases $a_1, \ldots, a_n$. The

$$
\begin{aligned}
e &::= n \mid x \mid e + e \mid e - e \mid \ldots \\
b &::= \texttt{tt} \mid \texttt{ff} \mid x < x \mid \ldots \\
s &::= \texttt{skip}^{\ell} \mid x =^{\ell} e \mid s \,\texttt{;}\, s \mid \texttt{if}\ b^{\ell}\ \{\,s\,\}\ \texttt{else}\ \{\,s\,\} \mid \texttt{for}\ b^{\ell}\ \{\,s\,\} \mid \texttt{select}^{\ell}\ \{\,a \ldots a\,\} \\
a &::= \texttt{case}\ x = \ \texttt{<-}^{\ell}\ ch\ \texttt{:}\ s \mid \texttt{case}\ ch\ \texttt{<-}^{\ell}\ e\ \texttt{:}\ s \\
p &::= (s\,\texttt{;}\,\texttt{select}^{\ell}\ \{\ \}) : \cdots : (s\,\texttt{;}\,\texttt{select}^{\ell}\ \{\ \})
\end{aligned}
$$

Fig. 2: BNF grammar of nano-Go

case $\texttt{case}\ x = \ \texttt{<-}\ ch\ \texttt{:}\ s$ reads a value from channel ch, stores it in the variable x, and proceeds to execute s. The case $\texttt{case}\ ch\ \texttt{<-}\ e\ \texttt{:}\ s$ writes the value of the expression e to channel ch and proceeds to execute s. Reading and writing messages is synchronous: a writing process blocks without an available receiver. Similarly a reading process cannot proceed until a writing process is ready to supply an input.

We assume that all statements and cases have been uniquely labeled. To be able to refer to specific labels occurring in a given statement or case we define the three functions *first*, *last*, and *labels* in Fig. 3. Each of these accept a labeled statement or case as input, *first* returns a label, whereas *last* and *labels* return a set of labels. For example, for the statement $s = \texttt{if}\ \texttt{tt}^{\ell_0}\ \{\,x =^{\ell_1} 1\,\}\ \texttt{else}\ \{\,\texttt{skip}^{\ell_2}\,\}$ we get $\mathit{first}(s) = \ell_0$ while $\mathit{last}(s) = \{\ell_1, \ell_2\}$ and $\mathit{labels}(s) = \{\ell_0, \ell_1, \ell_2\}$. Technically $\texttt{skip}^{\ell}$ is not a valid statement in concrete Go syntax, but we include it nevertheless as it is convenient (as the identity) in translating valid Go statement sequences into abstract syntax trees (ASTs) with only binary statement composition.

We provide an operational semantics of nano-Go in Fig. 4. In the semantics a system configuration consists of an ordered sequence of process configurations $c_1 \ldots c_n$. This setup can capture execution from the point just after all go-routines have been started. Each process configuration is a pair $c_i = \langle s_i, \rho_i \rangle$ where the store ρ_i captures the values of the ith process's variables and s_i is either a statement or a case (also denoted a_i) that captures the program point of the ith process. As traditional we express message-passing communication with annotation labels, writing $ch!v$ and $ch?v$ for a message write and a message read, respectively. Synchronization is expressed in rule SysComm by pairing a read with a write, whereas the rule SysTau expresses a

s / a	*first*	*last*	*labels*
$\texttt{skip}^{\ell}$	ℓ	$\{\ell\}$	$\{\ell\}$
$x =^{\ell} e$	ℓ	$\{\ell\}$	$\{\ell\}$
$s_1\ \texttt{;}\ s_2$	$\mathit{first}(s_1)$	$\mathit{last}(s_2)$	$\mathit{labels}(s_1) \cup \mathit{labels}(s_2)$
$\texttt{if}\ b^{\ell}\ \{\,s_1\,\}\ \texttt{else}\ \{\,s_2\,\}$	ℓ	$\mathit{last}(s_1) \cup \mathit{last}(s_2)$	$\{\ell\} \cup \mathit{labels}(s_1) \cup \mathit{labels}(s_2)$
$\texttt{for}\ b^{\ell}\ \{\,s\,\}$	ℓ	$\{\ell\}$	$\{\ell\} \cup \mathit{labels}(s)$
$\texttt{select}^{\ell}\ \{\,a_1 \ldots a_n\,\}$	ℓ	$\mathit{last}(a_1) \cup \cdots \cup \mathit{last}(a_n)$	$\{\ell\} \cup \mathit{labels}(a_1) \cup \cdots \cup \mathit{labels}(a_n)$
$\texttt{case}\ x = \ \texttt{<-}^{\ell}\ ch\ \texttt{:}\ s$	ℓ	$\mathit{last}(s)$	$\{\ell\} \cup \mathit{labels}(s)$
$\texttt{case}\ ch\ \texttt{<-}^{\ell}\ e\ \texttt{:}\ s$	ℓ	$\mathit{last}(s)$	$\{\ell\} \cup \mathit{labels}(s)$

Fig. 3: Definitions of *first*, *last*, and *labels*

$$\frac{}{\rho \vdash_{\mathcal{A}} n \Downarrow n}\ \textsc{Lit} \qquad \frac{}{\rho \vdash_{\mathcal{A}} x \Downarrow \rho(x)}\ \textsc{Var} \qquad \frac{\rho \vdash_{\mathcal{A}} e_1 \Downarrow v_1 \qquad \rho \vdash_{\mathcal{A}} e_2 \Downarrow v_2}{\rho \vdash_{\mathcal{A}} e_1 + e_2 \Downarrow v_1 + v_2}\ \textsc{Add}$$

$$\frac{\rho \vdash_{\mathcal{A}} e_1 \Downarrow v_1 \qquad \rho \vdash_{\mathcal{A}} e_2 \Downarrow v_2}{\rho \vdash_{\mathcal{A}} e_1 - e_2 \Downarrow v_1 - v_2}\ \textsc{Sub} \qquad \frac{}{\rho \vdash_{\mathcal{B}} \texttt{tt} \Downarrow \texttt{tt}}\ \textsc{True} \qquad \frac{}{\rho \vdash_{\mathcal{B}} \texttt{ff} \Downarrow \texttt{ff}}\ \textsc{False}$$

$$\frac{\rho(x_1) < \rho(x_2)}{\rho \vdash_{\mathcal{B}} x_1 \texttt{ < } x_2 \Downarrow \texttt{tt}}\ \textsc{LessThan1} \qquad \frac{\rho(x_1) \geq \rho(x_2)}{\rho \vdash_{\mathcal{B}} x_1 \texttt{ < } x_2 \Downarrow \texttt{ff}}\ \textsc{LessThan2}$$

$$\frac{}{\langle \texttt{skip}^{\ell}, \rho\rangle \xrightarrow{\tau} \rho}\ \textsc{Skip} \qquad \frac{\rho \vdash_{\mathcal{A}} e \Downarrow v}{\langle x \texttt{ =}^{\ell}\ e, \rho\rangle \xrightarrow{\tau} \rho[x \mapsto v]}\ \textsc{Assign}$$

$$\frac{\langle s_1, \rho\rangle \xrightarrow{\alpha} \langle s_3, \rho'\rangle}{\langle s_1 \texttt{ ; } s_2, \rho\rangle \xrightarrow{\alpha} \langle s_3 \texttt{ ; } s_2, \rho'\rangle}\ \textsc{Seq1} \qquad \frac{\langle s_1, \rho\rangle \xrightarrow{\alpha} \rho'}{\langle s_1 \texttt{ ; } s_2, \rho\rangle \xrightarrow{\alpha} \langle s_2, \rho'\rangle}\ \textsc{Seq2}$$

$$\frac{\rho \vdash_{\mathcal{B}} b \Downarrow \texttt{tt}}{\langle \texttt{if } b^{\ell}\ \{\, s_1 \,\}\ \texttt{else}\ \{\, s_2 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_1, \rho\rangle}\ \textsc{If1}$$

$$\frac{\rho \vdash_{\mathcal{B}} b \Downarrow \texttt{ff}}{\langle \texttt{if } b^{\ell}\ \{\, s_1 \,\}\ \texttt{else}\ \{\, s_2 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_2, \rho\rangle}\ \textsc{If2}$$

$$\frac{\rho \vdash_{\mathcal{B}} b \Downarrow \texttt{tt}}{\langle \texttt{for } b^{\ell}\ \{\, s_1 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_1 \texttt{ ; for } b^{\ell}\ \{\, s_1 \,\}, \rho\rangle}\ \textsc{For1} \qquad \frac{\rho \vdash_{\mathcal{B}} b \Downarrow \texttt{ff}}{\langle \texttt{for } b^{\ell}\ \{\, s_1 \,\}, \rho\rangle \xrightarrow{\tau} \rho}\ \textsc{For2}$$

$$\frac{\langle a_i, \rho\rangle \xrightarrow{\alpha} \langle s_i, \rho'\rangle}{\langle \texttt{select}^{\ell}\ \{\, a_1 \ldots a_n \,\}, \rho\rangle \xrightarrow{\alpha} \langle s_i, \rho'\rangle}\ \textsc{Select}$$

$$\frac{}{\langle \texttt{case } x \texttt{ = <-}^{\ell}\ ch \texttt{ : } s, \rho\rangle \xrightarrow{ch?v} \langle s, \rho[x \mapsto v]\rangle}\ \textsc{Read}$$

$$\frac{\rho \vdash_{\mathcal{A}} e \Downarrow v}{\langle \texttt{case } ch \texttt{ <-}^{\ell}\ e \texttt{ : } s, \rho\rangle \xrightarrow{ch!v} \langle s, \rho\rangle}\ \textsc{Write}$$

$$\frac{c_i \xrightarrow{\tau} c_i'}{c_1 \ldots c_i \ldots c_n \xRightarrow{i,\tau} c_1 \ldots c_i' \ldots c_n}\ \textsc{SysTau}$$

$$\frac{c_i \xrightarrow{ch!v} c_i' \qquad c_j \xrightarrow{ch?v} c_j' \qquad i \neq j}{c_1 \ldots c_i \ldots c_j \ldots c_n \xRightarrow{i,ch,v,j} c_1 \ldots c_i' \ldots c_j' \ldots c_n}\ \textsc{SysComm}$$

Fig. 4: Operational semantics of nano-Go

non-communicating action. We label the system-level transitions with the indices of the involved processes, writing i, τ for the ith process performing a non-communicating action and i, ch, v, j for process i writing a value v on channel ch which is read by process j. Following the (informal) semantics of Go, a process cannot send a message on a channel to itself. We model this restriction by testing the sender's index i against the receiver's index j. Because two senders can write to the same channel, in a given trace the semantics non-deterministically puts the message of one sender before another.

Nano-Go embodies two simplifying assumptions: there is no dynamic channel or process creation and message passing is synchronous. We are well aware of the limitations induced by these assumptions but find them orthogonal to the topic of this paper: process-local static analysis. As such we plan to address them in future work.

3 Background

We assume the reader is familiar with lattice theory [Grätzer, 1978, Davey and Priestley, 2002] and abstract interpretation [Cousot and Cousot, 1977, 1979], and only recall the more specialized and recent material on the abstract domain of lattice-valued regular expressions [Midtgaard et al., 2016b].

3.1 Lattice theory and abstract interpretation

An *atom* $a \in L$ is a lattice element such that if $\bot \sqsubseteq s \sqsubseteq a$ for some other $s \in L$ then $s = \bot$ or $s = a$. We write $Atoms(L)$ for L's set of atoms and let a, a' range over this set. An *atomic lattice* requires that for all non-bottom elements $s \in L$ there exists $a \in Atoms(L)$ such that $a \sqsubseteq s$. An *atomistic lattice* requires that each non-bottom element $s \in L$ is expressible as a join of atoms $s = \sqcup S$ for some $S \subseteq Atoms(L)$. An *atomistic Galois insertion* $\langle C; \sqsubseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A; \leq \rangle$ requires that α, γ connect two atomistic lattices such that $\alpha : Atoms(C) \longrightarrow Atoms(A)$ is surjective (α maps atoms to atoms and for all $a \in Atoms(A)$ there exists an $c \in Atoms(C)$ such that $\alpha(c) = a$).

3.2 Lattice-valued regular expressions

To analyze the network communication and content we will use the domain of *lattice-valued regular expressions* (LVREs) [Logozzo, 2004, Midtgaard et al., 2016b]. We recall here the basics of LVREs (sans complement as it is irrelevant for the problem at hand). Syntactically LVREs are regular expressions with its characters drawn from a lattice $\langle A; \sqsubseteq \rangle$:

$$\widehat{R}_A ::= \emptyset \mid \epsilon \mid \ell \mid \widehat{R}_A^* \mid \widehat{R}_A \cdot \widehat{R}_A \mid \widehat{R}_A + \widehat{R}_A \mid \widehat{R}_A \,\&\, \widehat{R}_A \quad \text{where } \ell \in A \setminus \{\bot\}$$

We assume that the meaning of the *lattice literals* (A's elements) are given by a Galois insertion $\langle \wp(C); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A; \sqsubseteq \rangle$ and that α maps atoms to atoms: $\alpha : Atoms(\wp(C)) \longrightarrow Atoms(A)$. These assumptions are liberal enough to allow many standard domains from the Galois connection framework (signs, parity, constant propagation, intervals, etc.). A number of consequences follow from these basic assumptions:

$$\begin{array}{lll}
\mathcal{L}(\emptyset) = \emptyset & \mathcal{L}(r^*) = \cup_{i\geq 0}\, \mathcal{L}(r)^i & \mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2) \\
\mathcal{L}(\epsilon) = \{\epsilon\} & \mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2) & \mathcal{L}(r_1 \,\&\, r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2) \\
\mathcal{L}(\ell) = \{c \mid c \in \gamma(\ell)\} & &
\end{array}$$

Fig. 5: The denotation of lattice-valued regular expressions

$$\begin{array}{ll}
\widehat{\mathcal{D}}_a(\emptyset) = \emptyset & \\
\widehat{\mathcal{D}}_a(\epsilon) = \emptyset & \widehat{\mathcal{D}}_a(r_1 \cdot r_2) = \begin{cases} \widehat{\mathcal{D}}_a(r_1) \cdot r_2 + \widehat{\mathcal{D}}_a(r_2) & \epsilon \mathrel{\underset{\approx}{\sqsubseteq}} r_1 \\ \widehat{\mathcal{D}}_a(r_1) \cdot r_2 & \epsilon \mathrel{\underset{\approx}{\not\sqsubseteq}} r_1 \end{cases} \\
\widehat{\mathcal{D}}_a(\ell) = \begin{cases} \epsilon & a \sqsubseteq \ell \\ \emptyset & a \not\sqsubseteq \ell \end{cases} & \widehat{\mathcal{D}}_a(r_1 + r_2) = \widehat{\mathcal{D}}_a(r_1) + \widehat{\mathcal{D}}_a(r_2) \\
\widehat{\mathcal{D}}_a(r^*) = \widehat{\mathcal{D}}_a(r) \cdot r^* & \widehat{\mathcal{D}}_a(r_1 \,\&\, r_2) = \widehat{\mathcal{D}}_a(r_1) \,\&\, \widehat{\mathcal{D}}_a(r_2)
\end{array}$$

Fig. 6: The Brzozowski derivative of lattice-valued regular expressions

A is a complete lattice, A is atomic, and A is atomistic. They also have the consequence that γ is strict ($\gamma(\bot) = \emptyset$), that $\alpha : \mathit{Atoms}(\wp(C)) \longrightarrow \mathit{Atoms}(A)$ is surjective (we have an atomistic Galois insertion), and that A's atoms have no overlapping meaning ($\forall a, a'.\, a \neq a' \implies \gamma(a) \cap \gamma(a') = \emptyset$) [Midtgaard et al., 2016b].

We give meaning to the LVREs relative to the γ of the given Galois insertion. The denotation is given in Fig. 5. Based on this denotation two LVREs r, r' are ordered language-wise: $r \mathrel{\underset{\approx}{\sqsubseteq}} r' \iff \mathcal{L}(r) \subseteq \mathcal{L}(r')$. This ordering constitutes only a pre-order as it fails anti-symmetry. To regain a partial order we consider LVREs up to language equivalence $\widehat{R}_A/_{\approx}$. The resulting quotient domain constitutes a lattice with binary least upper bounds $+$ and greatest lower bounds $\&$. It follows from the definition of $\mathcal{L}$ that, e.g., concatenation $\cdot$ is monotone in both arguments.

LVREs provide a number of domain operations: $\mathit{nullable} : \widehat{R}_A \longrightarrow \mathbb{B}$ determines whether the empty string is accepted by the language of a LVRE r ($\mathit{nullable}(r) \iff \epsilon \in \mathcal{L}(r)$). We omit the straight-forward, structural definition here for brevity. *The Brzozowski derivative* [Brzozowski, 1964] $\widehat{\mathcal{D}} : \mathit{Atoms}(A) \times \widehat{R}_A \longrightarrow \widehat{R}_a$ defined in Fig. 6 represents the language of a LVRE r remaining after having matched some $a \in \mathit{Atoms}(A)$ as the first character. One can prove that $\mathcal{L}(\widehat{\mathcal{D}}_a(r)) = \{w \mid \forall c \in \gamma(a).\, cw \in \mathcal{L}(r)\}$ for all $a \in \mathit{Atoms}(A)$ and $r \in \widehat{R}_A$. The definition of Brzozowski derivatives over LVREs extends structurally to strings: $\widehat{\mathcal{D}}_\epsilon(r) = r$ and $\widehat{\mathcal{D}}_{aw}(r) = \widehat{\mathcal{D}}_w(\widehat{\mathcal{D}}_a(r))$. Following Brzozowski [1964] derivatives can be used for translating LVREs to lattice-valued automata. One can thus view *LVREs as automata states* and the *derivatives as transitions*. A LVRE r is considered an accept state iff $\mathit{nullable}(r)$. This view is underlined by the fact that there are only a finite number of syntactically different LVRE derivatives (*corresponding to individual states*) up to associativity, commutativity, and idempotency (ACI) of $+$ when $\mathit{Atoms}(A)$ is finite.

In practice many derivatives are syntactically identical, e.g., over LVREs with intervals $\widehat{\mathcal{D}}_{[0;0]}([0;100] \cdot [1;2]^*) = \ldots = \widehat{\mathcal{D}}_{[100;100]}([0;100] \cdot [1;2]^*) = \epsilon \cdot [1;2]^*$ which motivated to group atoms with identical derivatives together in equivalence classes. For this purpose $\widehat{\mathit{range}}(r) : \widehat{R}_A \longrightarrow \widehat{\mathit{equiv}}_A$ computes a partition of $\mathit{Atoms}(A)$ such that two atoms a, a' are placed in the same equivalence class $a, a' \in [a''] \in \widehat{range}(r)$ if $\widehat{\mathcal{D}}_a(r) = \widehat{\mathcal{D}}_{a'}(r)$. Similarly $\widehat{overlay} : \widehat{equiv}_A \times \widehat{equiv}_A \longrightarrow \widehat{equiv}_A$ refines two partitions into a new partition coarser than both. $\widehat{overlay}$ is thus monotone over the lattice of partitions ordered under refinement [Grätzer, 1978]. Finally we require an operation $\widehat{repr} : (\wp(Atoms(A)) \setminus \{\emptyset\}) \longrightarrow Atoms(A)$ that returns a representative atom $a \in \widehat{repr}([a'])$ of a given equivalence class $[a']$ in a partition, and a second operation $\widehat{project} : (\wp(Atoms(A)) \setminus \{\emptyset\}) \longrightarrow A$ that returns a lattice element greater than all atoms in a given equivalence class: $\forall a \in [a'].\ a \sqsubseteq \widehat{project}([a'])$.

4 Shuffling lattice-valued regular expressions

To support analysis of arbitrary combinations of processes we extend LVREs with a symbolic shuffle operator. Formally we extend the grammar of LVREs with an additional production: $\widehat{R}_A ::= \ldots \ \mid\ \widehat{R}_A \parallel \widehat{R}_A$
Next we consider how to extend the various auxiliary operations to support the shuffle operator. First we define single string shuffling over the concrete domain C as follows:

$$\epsilon \parallel w \ = \ \{w\} \qquad\qquad w \parallel \epsilon \ = \ \{w\}$$
$$c_1w_1 \parallel c_2w_2 \ = \ \{c_1w \mid w \in w_1 \parallel c_2w_2\} \cup \{c_2w \mid w \in c_1w_1 \parallel w_2\}$$

This definition is taken from Sulzmann and Thiemann [2015]. For example, for $C = \{a, b, c\}$ we have $ab \parallel bc = \{abbc, abcb, babc, bacb, bcab\}$. The single string operation is commutative: for any strings w, w' we have $w \parallel w' = w' \parallel w$. We can lift the single string shuffling definition (elementwise) to languages (also from Sulzmann and Thiemann [2015]): $L_1 \parallel L_2 \ = \ \{w \mid w \in w_1 \parallel w_2 \wedge w_1 \in L_1 \wedge w_2 \in L_2\}$
Before we continue we establish a number of properties. Interestingly, the language shuffling operation is not idempotent. For example: $\{a\} \parallel \{a\} = \{aa\} \neq \{a\}$. We believe the following four properties are well known [Sulzmann and Thiemann, 2015] but nevertheless include them for completeness.

Lemma 1 (Shuffling of prefixed languages).

$$c_1 \cdot L_1 \parallel c_2 \cdot L_2 = c_1 \cdot (L_1 \parallel c_2 \cdot L_2) \ \cup \ c_2 \cdot (c_1 \cdot L_1 \parallel L_2)$$

Lemma 2 (Shuffling is commutative, distributive, associative).

$$L_1 \parallel L_2 = L_2 \parallel L_1 \qquad \text{(commutative)}$$
$$L \parallel (L_1 \cup L_2) = (L \parallel L_1) \cup (L \parallel L_2) \qquad \text{(distributive)}$$
$$L_1 \parallel (L_2 \parallel L_3) = (L_1 \parallel L_2) \parallel L_3 \qquad \text{(associative)}$$

We can prove a general shuffle property, that says that the shuffle of two arbitrary strings accounts for all possible splits of them: both the recursive shuffling of their first halves and their second halves are taken into consideration.

Lemma 3 (Generalized shuffle property).

$$\forall w_1, w_2, w_3, w_4 \in C^*.\ (w_1 \parallel w_2) \cdot (w_3 \parallel w_4) \subseteq (w_1 \cdot w_3) \parallel (w_2 \cdot w_4)$$

For example, by choosing $w_3 = \epsilon$ and $w_4 = c$ we obtain $\forall c \in C, w_1, w_2 \in C^*.\ (w_1 \parallel w_2) \cdot c \subseteq w_1 \parallel (w_2 \cdot c)$ which says that choosing c last is one possibility. Similarly in an alphabet with $\{rd, wr\} \subseteq C$ by choosing $w_3 = rd$ and $w_4 = wr$ as a corollary we obtain $\forall c \in C, w_1, w_2 \in C^*.\ (w_1 \parallel w_2) \cdot \{rd \cdot wr, wr \cdot rd\} \subseteq (w_1 \cdot rd) \parallel (w_2 \cdot wr)$.

Shuffling LVREs We can now give meaning to symbolic shuffling of LVREs as language shuffling of their meanings: $\mathcal{L}(r_1 \parallel r_2) = \mathcal{L}(r_1) \parallel \mathcal{L}(r_2)$. Consequently the symbolic operation is commutative and associative under language equality: $r_1 \parallel r_2 \approx r_2 \parallel r_1$ and $r_1 \parallel (r_2 \parallel r_3) \approx (r_1 \parallel r_2) \parallel r_3$. It is also monotone by definition: $r_1 \sqsubseteq_{\approx} r_1' \implies r_1 \parallel r_2 \sqsubseteq_{\approx} r_1' \parallel r_2$ (and similarly in the second argument by commutativity).

Derivatives and the nullable predicate Under the view of *expressions-as-states* and *derivatives-as-transitions*, the combined, synchronized automaton can take an a-step if either the first automaton can take an a-step or the second automaton can take an a-step. This leads to the following definition: $\widehat{\mathcal{D}}_a(r_1 \parallel r_2) = \widehat{\mathcal{D}}_a(r_1) \parallel r_2 \ + \ r_1 \parallel \widehat{\mathcal{D}}_a(r_2)$. Similarly the combined, shuffling automaton is in an acceptance state if both automata are in acceptance states. This leads to the following definition: $\mathit{nullable}(r_1 \parallel r_2) = \mathit{nullable}(r_1) \wedge \mathit{nullable}(r_2)$.
Our previous work established the Brzozowski equation for LVREs. We extend this result by showing how it also holds for LVREs with shuffle expressions:

Theorem 4 (Brzozowski's equation).

$$r \approx \sum_{a \in Atoms(A)} a\, \widehat{\mathcal{D}}_a(r) \ + \ \delta(r) \quad \textit{where} \ \ \delta(r) = \begin{cases} \epsilon & \epsilon \sqsubseteq_{\approx} r \\ \emptyset & \epsilon \not\sqsubseteq_{\approx} r \end{cases}$$

Based on this we can now extend the following lemmas to hold for LVREs with shuffle.

Lemma 5 (Meaning of derivatives). $\mathcal{L}(\widehat{\mathcal{D}}_a(r)) = \{w \mid \forall c \in \gamma(a).\, c \cdot w \in \mathcal{L}(r)\}$
Lemma 6 ($\widehat{\mathcal{D}}$ monotone in second argument). $r \sqsubseteq_{\approx} r' \implies \widehat{\mathcal{D}}_a(r) \sqsubseteq_{\approx} \widehat{\mathcal{D}}_a(r')$
Lemma 7 (Correctness of $\mathit{nullable}$). $\mathit{nullable}(r_1 \parallel r_2) \iff \epsilon \in \mathcal{L}(r_1 \parallel r_2)$

Finitely many derivatives We argue that for all r, there exists at most d_r different derivatives up to ACI of $+$. We first prove a syntactic characterization of all derivatives as a sum of derived shuffle pairs. There are only as many different derivatives (up to ACI of $+$) as there are different sets of such pairs. For each of the d_{r_1} different first components in such pairs there are at most d_{r_2} different second components and hence at most $d_{r_1} * d_{r_2}$ different pairs. This gives an upper bound of $2^{d_{r_1} * d_{r_2}}$ different sets of pairs. To reduce the number of derivatives further, we can utilize that $\parallel$ is commutative, meaning there are only as many unique derivative pairs as there are unique first and second components. This reduction is however not required to upper-bound the number of different derivatives.

The $\widehat{\mathit{range}}$ operator We extend the $\widehat{\mathit{range}}$ operator to shuffled expressions:

$$\widehat{\mathit{range}}(r_1 \parallel r_2) = \widehat{\mathit{overlay}}(\widehat{\mathit{range}}(r_1), \widehat{\mathit{range}}(r_2))$$

and we subsequently verify that this definition satisfies our formal requirements:

Lemma 8 ($\widehat{\mathit{range}}$ partitions atoms). $\forall r_1, r_2, [a_i] \in \widehat{\mathit{range}}(r_1 \parallel r_2), a, a' \in Atoms(A).$

$$a, a' \in [a_i] \implies \widehat{\mathcal{D}}_a(r_1 \parallel r_2) = \widehat{\mathcal{D}}_{a'}(r_1 \parallel r_2)$$

$$\begin{array}{ll} ch!\widehat{v} \in \widehat{Write}(\widehat{Val}) = Interval \times \{!\} \times \widehat{Val} & \widehat{\rho} \in \widehat{Store} = (Var \longrightarrow \widehat{Val})_\bot \\ ch?\widehat{v} \in \widehat{Read}(\widehat{Val}) = Interval \times \{?\} \times \widehat{Val} & \widehat{h}, \widehat{f} \in \widehat{R}_{\widehat{Ch}(\widehat{Val})} \\ \widehat{Ch}(\widehat{Val}) = \widehat{Write}(\widehat{Val}) \times \widehat{Read}(\widehat{Val}) & \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \in \widehat{Cache} = Labels \longrightarrow \widehat{Store} \times \widehat{R}_{\widehat{Ch}(\widehat{Val})} \end{array}$$

Fig. 7: Analysis domains

5 Analysis

Our core analysis is a standard imperative analysis over abstract stores $\widehat{\rho} \in \widehat{Store}$, e.g., with intervals. It requires auxiliary, monotone functions $\widehat{assign}$, $\widehat{\mathcal{A}}$, $\widehat{true}$, and $\widehat{false}$ which are standard and omitted for space reasons. We assume they satisfy the following:

Lemma 9 (Soundness of $\widehat{\mathcal{A}}$, $\widehat{assign}$, $\widehat{true}$, $\widehat{false}$ [Midtgaard et al., 2016a]).

$$\forall e \in E, \widehat{\rho} \in \widehat{Store}.\ \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \ \wedge\ \rho \vdash_{\mathcal{A}} e \Downarrow v\}) \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho})$$
$$\forall \widehat{\rho}, x, \widehat{v}.\ \alpha_{st}(\{\rho[x \mapsto v] \mid v \in \gamma_v(\widehat{v}) \ \wedge\ \rho \in \gamma_{st}(\widehat{\rho})\}) \ \dot{\sqsubseteq}\ \widehat{assign}(\widehat{\rho}, x, \widehat{v})$$
$$\forall b, \widehat{\rho}.\ \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} b \Downarrow \texttt{tt}\}) \ \dot{\sqsubseteq}\ \widehat{true}(b, \widehat{\rho})$$
$$\forall b, \widehat{\rho}.\ \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} b \Downarrow \texttt{ff}\}) \ \dot{\sqsubseteq}\ \widehat{false}(b, \widehat{\rho})$$

where $\dot{\sqsubseteq}$ is the pointwise lifting of the value ordering $\sqsubseteq$ and where the definitions of α_v, γ_v and α_{st}, γ_{st} are postponed to Sec. 6.

Rather than try to track the state of each individual process simultaneously which would lead to a combinatorial explosion, each process is approximated by its network interaction and analyzed in isolation against a given environment of network communication behaviour. We thus let LVREs of futures track *writes* and *reads* over a given channel when analyzing an individual process and set up a product $\widehat{Ch}(\widehat{Val})$ of a write domain ($\widehat{Write}(\widehat{Val})$ in Fig. 7 captures approximate write characters) and a read domain ($\widehat{Read}(\widehat{Val})$ in Fig. 7 captures approximate read characters).[3] We use an interval in both to capture channel numbers. The analysis future $\widehat{f} \in \widehat{R}_{\widehat{Ch}(\widehat{Val})}$ represents the network communication the surrounding environment may offer. Finally the analysis specification is expressed as two global analysis caches $\widehat{\mathcal{E}}$, $\widehat{\mathcal{X}}$ where $\widehat{\mathcal{E}}(\ell) = (\widehat{\rho}, \widehat{f})$ capture the store and future upon entry to the statement labeled ℓ and $\widehat{\mathcal{X}}(\ell)$ capture a corresponding pair upon completion of the statement. The caches are naturally partitioned into process-individual parts $\widehat{\mathcal{E}^1}, \ldots, \widehat{\mathcal{E}^n}$ with $dom(\widehat{\mathcal{E}^i}) = labels(s_i)$ such that $\widehat{\mathcal{E}^i}$ accounts for the labels in process i's body s_i (and similarly for $\widehat{\mathcal{X}^i}$). Collectively these are non-overlapping and span *Labels* for an entire program. Notationally we write $\widehat{\mathcal{E}^i_\rho}(\ell)$ and $\widehat{\mathcal{E}^i_f}(\ell)$ to refer to the two components of $\widehat{\mathcal{E}^i}(\ell)$ (and similarly for $\widehat{\mathcal{X}^i}$).

5.1 Analysis algorithm

The analysis is structured in two parts: an intra-process part (in Fig. 8 and Fig. 9) for analyzing each individual process in isolation and an inter-process part (in Fig. 10) for analyzing a system of processes with the latter depending on the former.

[3] The product with singleton sets $\{!\}$ and $\{?\}$ is just presentational: one component denotes writes and another component denotes reads.

The intra-process analysis specification in Fig. 8 is standard [Nielson et al., 1999] modulo the cases for network interaction. Here a read action involves a suitable derivative of the future wrt. a write action (and vice versa). The specification is slightly complicated by our partitioning of atoms into equivalence classes with identical derivatives. Algorithmically we use this intra-process analysis to infer process-local caches $\widehat{\mathcal{E}^i}$ and $\widehat{\mathcal{X}^i}$ for a given initial future $\widehat{f}$ and statement s_i.

Given an acceptable analysis result $\widehat{\mathcal{E}^i}$ and $\widehat{\mathcal{X}^i}$ of a process s_i we subsequently use $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i)$ in Fig. 9 to read off the collective network communication history of this process's writes and reads. $\mathcal{H}$ returns a pair of two languages: The first component denotes the *prefix* p of network communication strings that may arise from a statement s_i, whereas the second component denotes the *complete* language c of network communication strings that may arise from an end-to-end execution of statement s_i. Collectively $p + c$ represents all prefixes of s_i's network communication. For a less structured language we expect Tarjan's algorithm [Tarjan, 1981] could be adapted.

We can now combine intra-process communication histories $\langle p_i, c_i \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i)$ via the shuffle operator to obtain a better approximation of futures and repeat the intra-process analysis from this new starting point. For example, for an analysis of three processes s_1, s_2, s_3 we reanalyze s_1 under the future $\widehat{\mathcal{E}^1_f}(\mathit{first}(s_1))\ \&\ (p_2{+}c_2) \parallel (p_3{+}c_3)$. To soundly model how a third party process may interfere or communicate with either party before or after a message synchronization the inter-process analysis specification in Fig. 10 imposes a closure requirement. In this setup a future *write* followed by a matching *read* (and vice versa) may match up and thereby cancel each other out. We express this requirement with derivatives: a *write* requires a derivative with respect to a suitable *read* (and vice versa). Since $\widehat{\mathit{range}}$ groups into equivalence classes atoms with identical derivatives, a little extra care is needed to find equivalence classes for which two consecutive derivatives are guaranteed to yield the same. This is the purpose of the bottom requirement in Fig. 10, which utilizes that the atoms of $\widehat{Ch}(\widehat{Val})$ can be partitioned with a pair (the first projection π_1 partitions the atoms $\mathit{Atoms}(\widehat{\mathit{Write}}(\widehat{Val})) \times \{\bot\}$ and the second projection π_2 the atoms $\{\bot\} \times \mathit{Atoms}(\widehat{\mathit{Read}}(\widehat{Val}))$).

6 Soundness

The soundness proof is complicated by the fact that we relate two concepts of inherently different shape: we approximate a property expressible as a set of (prefix) traces, albeit where a single computation step in the trace itself may require a *deriviation tree* in the structural operational semantics of the corresponding process, whereas we specify the static analysis as a syntax-directed *acceptability relation* over the program text of each participating process. We proceed by first proving local statement-level soundness and then use this to prove system-level soundness. As these assume some over-approximate futures, we finally prove how an acceptable analysis result may be combined into a better over-approximation.

The analysis is parametric in the value abstraction, assuming it is given as an atomistic Galois insertion $\wp(\mathit{Val}) \underset{\alpha_v}{\overset{\gamma_v}{\leftrightarrows}} \widehat{Val}$. The value abstraction is straightforwardly

$$\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{skip}^\ell \text{ iff } \widehat{\mathcal{E}^i}(\ell) \sqsubseteq \widehat{\mathcal{X}^i}(\ell)$$

$$\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models x =^\ell e \text{ iff } (\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(e, \widehat{\rho})), \widehat{f}) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \text{ where } (\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$$

$$\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 \texttt{ ; } s_2 \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 \wedge \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_2 \wedge \forall \ell_1 \in last(s_1).\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{E}^i}(first(s_2))$$

$$\begin{aligned}
&\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{if } b^\ell \, \{ s_1 \} \texttt{ else } \{ s_2 \} \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 \wedge \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_2 \wedge \\
&\quad (\widehat{true}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(first(s_1)) \wedge (\widehat{false}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(first(s_2)) \wedge \\
&\quad \forall \ell_1 \in last(s_1),\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \wedge \forall \ell_2 \in last(s_2).\ \widehat{\mathcal{X}^i}(\ell_2) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \\
&\quad \text{where } (\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)
\end{aligned}$$

$$\begin{aligned}
&\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{for } b^\ell \, \{ s_1 \} \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 \wedge \\
&\quad (\widehat{true}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(first(s_1)) \wedge (\widehat{false}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \wedge \\
&\quad \forall \ell_1 \in last(s_1).\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{E}^i}(\ell) \text{ where } (\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)
\end{aligned}$$

$$\begin{aligned}
&\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{select}^\ell \, \{ a_1 \ldots a_n \} \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models a_1 \wedge \ \ldots \ \wedge \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models a_n \wedge \\
&\quad \widehat{\mathcal{E}^i}(\ell) \sqsubseteq \widehat{\mathcal{E}^i}(first(a_1)) \wedge \ \ldots \ \wedge \widehat{\mathcal{E}^i}(\ell) \sqsubseteq \widehat{\mathcal{E}^i}(first(a_n)) \wedge \\
&\quad \forall \ell_1 \in last(a_1).\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \wedge \ \ldots \ \wedge \forall \ell_n \in last(a_n).\ \widehat{\mathcal{X}^i}(\ell_n) \sqsubseteq \widehat{\mathcal{X}^i}(\ell)
\end{aligned}$$

$$\begin{aligned}
&\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{case } x \texttt{ = <-}^\ell \, ch \texttt{ : } s \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s \wedge \\
&\quad \forall [ch!\widehat{v}_a] \in \widehat{range}(\widehat{f}). \\
&\qquad (ch!\widehat{v} = \widehat{project}([ch!\widehat{v}_a]) \wedge \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a])}(\widehat{f}) \not\sqsubseteq \emptyset \\
&\qquad \implies (\widehat{assign}(\widehat{\rho}, x, \widehat{v}), \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a])}(\widehat{f})) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \sqsubseteq \widehat{\mathcal{E}^i}(first(s))) \\
&\quad \text{where } (\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)
\end{aligned}$$

$$\begin{aligned}
&\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{case } ch \texttt{ <-}^\ell \, e \texttt{ : } s \text{ iff } \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s \wedge \\
&\quad \forall [ch?\widehat{v}_a] \in \widehat{range}(\widehat{f}). \\
&\qquad (ch?\widehat{v} = \widehat{project}([ch?\widehat{v}_a]) \wedge \widehat{v} \sqcap \widehat{v}' \neq \bot \wedge \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{f}) \not\sqsubseteq \emptyset \\
&\qquad \implies (\widehat{\rho}, \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{f})) \sqsubseteq \widehat{\mathcal{X}^i}(\ell) \sqsubseteq \widehat{\mathcal{E}^i}(first(s))) \\
&\quad \text{where } (\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell) \wedge \widehat{v}' = \widehat{\mathcal{A}}(e, \widehat{\rho})
\end{aligned}$$

Fig. 8: Intra-process analysis specification

lifted to a Galois insertion over stores: $\wp(Var \hookrightarrow Val) \underset{\alpha_{st}}{\overset{\gamma_{st}}{\leftrightarrows}} \widehat{Store}$. Finally the channel abstraction $\wp(Action) \underset{\alpha_{ch}}{\overset{\gamma_{ch}}{\leftrightarrows}} \widehat{Ch}(\widehat{Val})$ is a standard Cartesian abstraction with $Action = WrAction \cup RdAction$, $\alpha_{ch}(S) = (\alpha_{wr}(\{ch!v \in S\}), \alpha_{rd}(\{ch?v \in S\}))$ and $\gamma_{ch}(\widehat{v}_w, \widehat{v}_r) = \gamma_{wr}(\widehat{v}_w) \cup \gamma_{rd}(\widehat{v}_r)$. We sometimes abbreviate $\alpha_{ch}(S)$ as $\widehat{S}$. The channel abstraction itself utilizes two atomistic Galois insertions $\wp(WrAction) \underset{\alpha_{wr}}{\overset{\gamma_{wr}}{\leftrightarrows}} \widehat{Write}(\widehat{Val})$ with $\alpha_{wr}(S) = \bigsqcup_{ch!v \in S}(\alpha_{Int}(\{ch\}), \alpha_v(\{v\}))$ and $\gamma_{wr}([l;u], \widehat{v}) = \bigcup_{\substack{ch \in \gamma_{Int}([l;u]) \\ v \in \gamma_v(\widehat{v})}} \{ch!v\}$ and similarly for α_{rd}, γ_{rd} [Midtgaard et al., 2016a].

6.1 Statement-level soundness

The following two lemmas express soundness at the statement level for both SOS steps leading to a terminal and a non-terminal configuration. Properties related to how fu-

$$
\begin{aligned}
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{skip}^\ell) &= \langle \epsilon, \epsilon \rangle \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, x =^\ell e) &= \langle \epsilon, \epsilon \rangle \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1 \texttt{ ; } s_2) &= \langle p_1 + (c_1 \cdot p_2), c_1 \cdot c_2 \rangle \\
&\quad \text{where } \langle p_1, c_1 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1) \text{ and } \langle p_1, c_2 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_2) \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{if } b^\ell \,\{\, s_1 \,\}\texttt{ else }\{\, s_2 \,\}) &= \langle p_1 + p_2, c_1 + c_2 \rangle \\
&\quad \text{where } \langle p_1, c_1 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1) \text{ and } \langle p_1, c_2 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_2) \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{for } b^\ell \,\{\, s \,\}) &= \langle c^* \cdot p, c^* \rangle \text{ where } \langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s) \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{select}^\ell \,\{\, a_1 \dots a_n \,\}) &= \langle \epsilon + \sum_i p_i, \sum_i c_i \rangle \\
&\quad \text{where } \langle p_i, c_i \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, a_i) \text{ and } 1 \le i \le n \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{case } x \texttt{ = <-}^\ell \, ch \texttt{ : } s) &= \langle \epsilon + ch?\widehat{v} + ch?\widehat{v} \cdot p, ch?\widehat{v} \cdot c \rangle \\
&\quad \text{where } \widehat{v} = \widehat{\mathcal{E}^i_\rho}(\mathit{first}(s))(x) \text{ and } \langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s) \\
\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{case } ch \texttt{ <-}^\ell \, e \texttt{ : } s) &= \langle \epsilon + ch!\widehat{v} + ch!\widehat{v} \cdot p, ch!\widehat{v} \cdot c \rangle \\
&\quad \text{where } \widehat{v} = \widehat{\mathcal{A}}(e, \widehat{\mathcal{E}^i_\rho}(\ell)) \text{ and } \langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)
\end{aligned}
$$

Fig. 9: Reading off a collective trace history

$$
\begin{aligned}
&\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1 : \dots : s_n \text{ iff } \forall i.\ \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_i \ \wedge \\
&\qquad \forall i, \ell, [ch!\widehat{v}] \in \widehat{\mathit{overlay}}(\,\{\pi_1(\widehat{\mathit{range}}(\widehat{\mathcal{E}^i_f}(\ell)))\} \cup \bigcup_{[ch'!\widehat{v}'] \in \widehat{\mathit{range}}(\widehat{\mathcal{E}^i_f}(\ell))} \pi_2(\widehat{\mathit{range}}(\widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([ch'!\widehat{v}'])}(\widehat{\mathcal{E}^i_f}(\ell))))\,). \\
&\qquad\qquad \widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([ch!\widehat{v}])\widehat{\mathit{repr}}([ch?\widehat{v}])}(\widehat{\mathcal{E}^i_f}(\ell)) \underset{\sim}{\sqsubset} \widehat{\mathcal{E}^i_f}(\ell) \ \wedge \\
&\qquad \forall i, \ell, [ch?\widehat{v}] \in \widehat{\mathit{overlay}}(\,\{\pi_2(\widehat{\mathit{range}}(\widehat{\mathcal{E}^i_f}(\ell)))\} \cup \bigcup_{[ch'?\widehat{v}'] \in \widehat{\mathit{range}}(\widehat{\mathcal{E}^i_f}(\ell))} \pi_1(\widehat{\mathit{range}}(\widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([ch'?\widehat{v}'])}(\widehat{\mathcal{E}^i_f}(\ell))))\,). \\
&\qquad\qquad \widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([ch?\widehat{v}])\widehat{\mathit{repr}}([ch!\widehat{v}])}(\widehat{\mathcal{E}^i_f}(\ell)) \underset{\sim}{\sqsubset} \widehat{\mathcal{E}^i_f}(\ell)
\end{aligned}
$$

Fig. 10: Inter-process analysis specification

tures propagate across processes are handled at the system level. The two lemmas are reminiscent of lemmas 7.9, 7.10 in our previous work [Midtgaard et al., 2016a] with the key difference that those were expressed in terms of an instrumented semantics. Both of these lemmas express soundness of a network action α against the environment using a derivative of the converse action $\overline{\alpha}$ defined as $\overline{\tau} = \epsilon \quad \overline{ch?v} = ch!v \quad \overline{ch!v} = ch?v$.

Lemma 10 (One step statement soundness, terminal). *If* $\langle s, \rho \rangle \xrightarrow{\alpha} \rho'$*,* $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s$*, and* $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s)))$ *then* $\forall \ell \in \mathit{last}(s).\ \ \rho' \in \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell)) \ \wedge\ \widehat{\mathcal{D}}_{\widehat{\overline{\alpha}}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) \underset{\sim}{\sqsubset} \widehat{\mathcal{X}^i_f}(\ell)$

Lemma 11 (One step statement soundness, non-terminal). *If* $\langle s, \rho \rangle \xrightarrow{\alpha} \langle s', \rho' \rangle$*,* $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s$*,* $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s)))$*, and* $\widehat{\mathcal{D}}_{\widehat{\overline{\alpha}}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) \not\underset{\sim}{\sqsubset} \emptyset$ *then* $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s' \ \wedge\ \rho' \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s'))) \ \wedge\ \widehat{\mathcal{D}}_{\widehat{\overline{\alpha}}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) \underset{\sim}{\sqsubset} \widehat{\mathcal{E}^i_f}(\mathit{first}(s'))$

6.2 System-level soundness

To express system-level soundness we introduce two homomorphisms over the labels of the semantics's system-level transitions:

$$\hbar_k(i,\tau) = \epsilon \qquad\qquad f_k(i,\tau) = \epsilon$$

$$\hbar_k(i,ch,v,j) = \begin{cases} ch!v & k=i \\ ch?v & k=j \\ \epsilon & k \notin \{i,j\} \end{cases} \qquad f_k(i,ch,v,j) = \begin{cases} ch?v & k=i \\ ch!v & k=j \\ ch!v \cdot ch?v & k \notin \{i,j\}, i<j \\ ch?v \cdot ch!v & k \notin \{i,j\}, i>j \end{cases}$$

Note how in two cases f_k maps a single communication to a string of two characters: write-read or read-write, depending on the index of the participant (we have chosen somewhat arbitrarily to let the lowest process index go first).

Theorem 12 (Analysis soundness). *For all programs $s_1 : \cdots : s_n$, initial stores ρ_{init}, acceptable analysis answers $\widehat{\mathcal{E}}, \widehat{\mathcal{X}}$ such that $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1 : \cdots : s_n$ and the initial store is soundly account for $\forall i.\ \rho_{init} \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_i)))$, and arbitrary traces $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ with futures soundly accounted for $\forall i.\ f_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$ then for any i such that $1 \le i \le n$ and $c'_i = \langle s'_i, \rho'_i\rangle$ we have $\rho'_i \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s'_i))) \wedge \widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_k)}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))$*

Intuitively, the analysis accounts for all execution traces in the program such that the abstract store associated to each entry accounts for the reachable concrete stores and the abstract future associated to each entry accounts for the network communication of the surrounding process environment. We prove the generalization that concludes $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s'_i$ in addition to the above.

6.3 Soundness of iterative approach

The above proves soundness of the process analysis assuming that all futures are soundly accounted for in the initial statements of the individual processes, e.g., from worst-case assumptions $\forall i.\ \widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)) = \top^*$. To do better, we first express futures as a suitable shuffling of histories:

Lemma 13 (Futures as histories, sans sum). *For all programs $s_1 : \cdots : s_n$, initial stores ρ_{init}, and traces $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ such that for all $1 \le i \le n$ and $c'_i = \langle s'_i, \rho'_i\rangle$ we have $f_i(\alpha_1 \ldots \alpha_k) \in \big\|_{j \neq i} \hbar_j(\alpha_1 \ldots \alpha_k)$*

As a corollary by monotonicity of $\|$ we obtain the following:

Corollary 14 (Futures as histories, with sum). *For all programs $s_1 : \cdots : s_n$, initial stores ρ_{init}, and traces $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ such that for all $1 \le i \le n$ and $c'_i = \langle s'_i, \rho'_i\rangle$ we have $f_i(\alpha_1 \ldots \alpha_k) \in \big\|_{j \neq i} \left(\sum_{k' \le k} \hbar_j(\alpha_1 \ldots \alpha_{k'})\right)$*

Finally we can prove soundness of $\mathcal{H}$ from an acceptable analysis result:

Lemma 15 (History soundness). *For all programs $s_1 : \cdots : s_n$, initial stores ρ_{init}, and traces $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ such that for all $1 \le i \le n$ and $c'_i = \langle s'_i, \rho'_i\rangle$ and analysis answers $\widehat{\mathcal{E}}, \widehat{\mathcal{X}}$ such that $\rho_{init} \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_i)))$, $f_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$, and $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_i$. we have $\hbar_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(p + c)$ where $\langle p, c\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$*

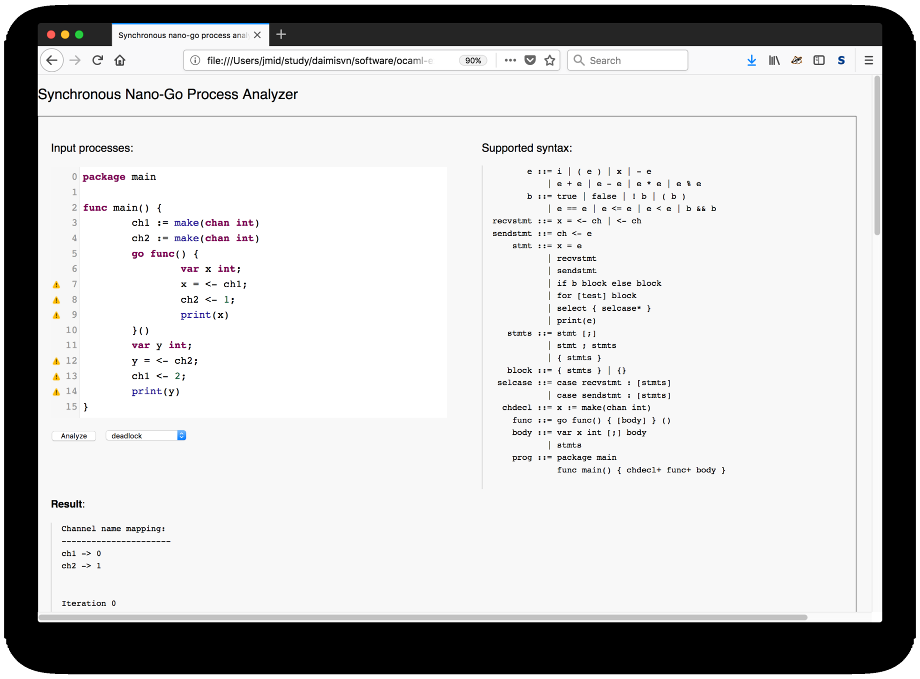
From a sound analysis result we utilize Corollary 14, Theorem 15, and monotonicity of $\|$ to obtain a (potentially better) approximation of the futures which proves the soundness of the inter-process analysis result shuffling:

$$f_i(\alpha_1 \ldots \alpha_k) \in \Big\|_{j \neq i} \left(\sum_{k' \le k} \hbar_j(\alpha_1 \ldots \alpha_{k'}) \right) \subseteq \Big\|_{\substack{j \neq i \\ \langle p_j, c_j\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)}} \mathcal{L}(p_j + c_j)$$

7 Implementation

To illustrate feasibility of our approach we have implemented a proof-of-concept prototype in OCaml. The prototype takes roughly 4200 lines of code and is available for download at `https://github.com/jmid/nano-go`. It is structured as a traditional front end with a lexer and a parser. The input is subsequently translated and labeled into an internal AST representation. The analysis walks this AST repeatedly until stabilization. As the shuffling operator over LVREs is commutative and associative we represent a sequence of shuffles $r_1 \parallel (r_2 \parallel (\cdots \parallel r_n))$ internally as a sorted sequence, since the element order does not matter. Since $\mathcal{L}(\emptyset \parallel r) = \mathcal{L}(\emptyset)$ and $\mathcal{L}(\epsilon \parallel r) = \mathcal{L}(r)$ we furthermore simplify LVREs internally from the former to the latter. Such meaning-preserving simplifications are common in derivative-based language processors [Owens et al., 2009]. We have implemented the closure requirement from the inter-process analysis specification in Fig. 10 as a local iteration, that repeats an inclusion of consecutive reads-and-writes (and vice versa) until stabilization. As there are only finitely many derivatives of a given future this iteration is bound to terminate. We only trigger the closure iteration on newly formed entries. Internally in the intra-process analysis the prototype widens on loop headers to ensure termination. Seen as a black box, the intra-process analysis is a deterministic function expecting a future $\widehat{f}$ as input. Since there are only finitely many derivatives of a given $\widehat{f}$ we do not need to widen over futures. Finally we widen over abstract stores by pointwise lifting of a traditional interval widening operator [Cousot and Cousot, 1976]. In the outer inter-process analysis the prototype starts from a safe $\top^*$ approximation of futures and runs at most 100 iterations of the inter-process analysis to improve on this worst case assumption.

We have used the `js_of_ocaml` compiler to create a client-side web-interface for the prototype, available at `https://jmid.github.io/nano-go/`. To illustrate the applicability of the analysis we have implemented two kinds of warnings based on the analysis results: We mark a statement s^ℓ with $\widehat{\mathcal{E}}^i_\rho(\ell) = \bot$ as *unreachable* and read and write actions with an empty derivative over futures as *unable to succeed*. Both of these are safety properties compatible with the analysis output. Fig. 11 illustrates these


Synchronous nano-go process anal
file:///Users/jmid/study/daimisvn/software/ocaml-e
90%
Search
Analyze
deadlock

Synchronous Nano-Go Process Analyzer

Input processes:

```
 0 package main
 1
 2 func main() {
 3         ch1 := make(chan int)
 4         ch2 := make(chan int)
 5         go func() {
 6                 var x int;
 7                 x = <- ch1;
 8                 ch2 <- 1;
 9                 print(x)
10         }()
11         var y int;
12         y = <- ch2;
13         ch1 <- 2;
14         print(y)
15 }
```

Supported syntax:

```
       e ::= i | ( e ) | x | - e
           | e + e | e - e | e * e | e % e
       b ::= true | false | ! b | ( b )
           | e == e | e <= e | e < e | b && b
recvstmt ::= x = <- ch | <- ch
sendstmt ::= ch <- e
    stmt ::= x = e
           | recvstmt
           | sendstmt
           | if b block else block
           | for [test] block
           | select { selcase* }
           | print(e)
   stmts ::= stmt [;]
           | stmt ; stmts
           | { stmts }
   block ::= { stmts } | {}
 selcase ::= case recvstmt : [stmts]
           | case sendstmt : [stmts]
  chdecl ::= x := make(chan int)
    func ::= go func() { [body] } ()
    body ::= var x int [;] body
           | stmts
    prog ::= package main
             func main() { chdecl+ func+ body }
```

Result:

```
Channel name mapping:
---------------------
ch1 -> 0
ch2 -> 1


Iteration 0
```

Fig. 11: Screenshot of the prototype's web-interface

warnings in the web-interface on a simple deadlock example with two processes both attempting to read before writing, thereby mutually blocking each other. In the example, the prototype highlights the read statements in lines 7 and 12 as unable to succeed and the subsequent lines as unreachable.

For a more elaborate example, consider the nano-Go program in Fig. 12 ported from Stadtmüller et al. [2016]. The program declares two channels `ch` and `done` and consists of 5 processes. The first process (`Send`) in line 6 sends an integer over channel `ch` and thereby triggers one of two competing receiver processes (`Recv1` and `Recv2`) in lines 7 and 12. The successful receiver acknowledges reception by subsequently writing the received value on channel `done`. A fourth process (`Work`) in line 17 simply runs an infinite loop, while the main process at the end expects to receive two acknowledgments. In the first inter-process iteration the intra-process analysis infers the history $\epsilon + \texttt{ch}![42;42]$ for `Send`, $\epsilon + \texttt{ch}?[-\infty;+\infty] + \texttt{ch}?[-\infty;+\infty] \cdot \texttt{done}![-\infty;+\infty]$ for `Recv1` and `Recv2`, ϵ for `Work`, and $\epsilon + \texttt{done}?[-\infty;+\infty] + \texttt{done}?[-\infty;+\infty] \cdot \texttt{done}?[-\infty;+\infty]$ for the final process. Each of these are obtained from the worst case assumption $\top^*$ about futures. Throughout the remaining inter-process iterations the results for `Send` and `Work` are unchanged. In the second inter-process iteration when the above histories are shuffled and fed to an intra-process re-analysis, `Recv1`'s and `Recv2`'s histories are both improved to $\epsilon + \texttt{ch}?[42;42] + \texttt{ch}?[42;42] \cdot \texttt{done}![42;42]$ and the final process's history is improved to $\epsilon + \texttt{done}?[-\infty;+\infty]$. In the third iteration `Recv1`'s and `Recv2`'s histories remain unchanged while the final process's history is improved to $\epsilon + \texttt{done}?[42;42]$. The fourth

```
 1  package main
 2
 3  func main() {
 4      ch := make(chan int)
 5      done := make(chan int)
 6      go func() { ch <- 42 }() // Send
 7      go func() {              // Recv1
 8          var val int;
 9          val = <-ch; done <- val;
10      }()
11      go func() {              // Recv2
12          var val int;
13          val = <-ch; done <- val;
14      }()
15      go func() { for {} }()   // Work
16      <-done;
17      <-done
18  }
```

Fig. 12: A deadlock example ported from Stadtmüller et al. [2016]

and final iteration confirms inter-process stabilization. The analysis prototype thereby discovers that the second read statement in line 17 is unable to succeed.

Table 1 lists performance of the command-line prototype on a number of examples, including two additional example programs ported from Stadtmüller et al. [2016]. The reported timings were measured using the `time` tool for the natively compiled prototype running on a lightly loaded 3.1Ghz MacBook Pro laptop. For each program we list the number of processes and channels, the number of inter-process analysis iterations, and the minimum, maximum, and average analysis time across five analysis runs. Whereas these numbers are promising they are also preliminary and included here only to demonstrate feasibility of the approach. The deadlock examples from Fig. 11 and 12 illustrate how it is possible to catch some deadlocks despite analyzing a safety property over-approximately. In contrast, our tool raises no warnings when analyzing the `philo` dining philosophers program listed in Table 1 as it *may* execute successfully. In Sec.8 we further compare our approach with that of Stadtmüller et al. [2016].

In order to meet our long term goal of scalable inter-process analysis, we expect a number of optimizations to be relevant. For one, an alternative implementation based on extracting constraints would only need to traverse the AST once to eliminate the repeated interpretive overhead. For another, one could consider caching (or dependencies between) the intra-process analysis results to avoid needless intra-process reanalysis. Finally, our division into repeated intra-process analysis lends itself to parallelization.

program	# proc.	# chan.	# interproc. iter.	min time	max time	avg.time
initial example, Fig. 1	3	2	4	0.014	0.018	0.0158
simple deadlock, Fig. 11	2	2	3	0.010	0.012	0.0110
deadlock, Fig. 12	5	2	4	0.055	0.058	0.0572
fanIn	4	3	4	0.896	0.938	0.9140
philo	4	1	3	0.770	0.793	0.7822

Table 1: Preliminary performance measurement (all reported times are in seconds)

8 Related work

Historically, channel-based concurrency in the style of Hoare's CSP has influenced programming languages such as Concurrent ML (CML) [Reppy, 1999] and more recently Google's Go programming language. Static analysis of CSP-like programs dates back to an early application of abstract interpretation [Cousot and Cousot, 1980], a *whole program* analysis. Since the nineties various forms of static analysis of concurrent programs have been investigated. In an early contribution Mercouroff [1991] developed an abstract interpretation-based, polynomial-time analysis of CSP-like programs. It could infer the communication count on each channel connecting two processes. Nielson and Nielson [1994] developed a type and effect system for CML with dynamic process and channel creation that could predict, e.g., the number of processes and channels created during a program's execution. Compared to our analysis it did not characterize the content of the messages sent. Colby [1995] subsequently developed an abstract interpretation of CML, including dynamic process creation. Akin to Nielson and Nielson [1994] he analyzed the communication topology of a given program, to answer questions of the form *'which occurrences of* `receive` *can a* `transmit` *occurrence reach'?* In subsequent work various analyses for process calculi were investigated. For example, Venet [1998] developed a framework for static analysis of π-calculus programs, Rydhof Hansen et al. [1999] developed a static analyses for control flow and occurrence counting of mobile ambients, and Feret [2000] developed control-flow and occurrence counting analyses of π-calculus programs.

Kobayashi and co-authors have since developed a range of type-based static analyses for π-calculus: Igarashi and Kobayashi [1997] developed a type-based analysis of channel communication count, Kobayashi [2005] developed an type-based information flow analysis including a type inference algorithm, Kobayashi [2006] developed a type system that guarantees deadlock-freedom including an type inference algorithm, and Kobayashi and Sangiorgi [2008] developed a hybrid *lock-freedom analysis* guaranteeing that certain communications will succeed while itself relying on deadlock-freedom and termination analyses. Most recently Giachino et al. [2014] have developed a refinement of Kobayashi's earlier deadlock-freedom analysis that can precisely detect deadlocks in value-passing CCS (and pi-calculus) programs with arbitrary numbers of processes while still permitting type inference. Since many of the process analyses can themselves be viewed as operating over a program abstraction (a process calculus term), they are inherently limited by the precision of this abstraction. Our work instead builds on a reduced product, in which information about program variables can influence the knowledge of network communication content and vice versa.

One may view our analysis analysis as an effect system specialized to inferring histories of synchronous network communication akin to Skalka et al. [2008] with the LVREs representing sets of traces of such events. In comparison to Skalka et al. [2008] our approach however also infers more precise information about the value of individual events: in that sense it refines the primitive notion of an event to a lattice value.

A number of recent papers develop static analyses for various subsets of Go. Ng and Yoshida [2016] first developed a static deadlock detection system for a subset of Go with a fixed number of processes and synchronous communication. Stadtmüller et al. [2016] then developed a trace-based deadlock analysis of *Synchronous Mini-Go*, a syntactically slightly bigger language than nano-Go. It built on earlier work by Sulzmann and Thiemann [2016] by first extracting regular expressions extended with *forkable behaviours* and subsequently analyzing these for deadlocks. Technically this involved both shuffling for the denotation of forkable behaviours and Brzozowski derivatives for the subsequent analysis. Recently Lange et al. [2017] have developed a verification framework for a bigger subset of Go, supporting both asynchronous message passing and recursion. It works by approximating program behaviours by *behavioural types* and a subsequent *bounded verification* of these. The above are primarily analyses for detecting potential deadlocks which our approach is not particularly geared for. However our value analysis is more precise since it utilizes a finer value abstraction than types. Botbol et al. [2017] develop a whole-program approach based on *lattice automata* [Le Gall and Jeannet, 2007] and *symbolic transducers* to analyze synchronous processes communicating via message passing and illustrate it with an application to MPI in C.

Miné [2014] developed a thread-modular analysis approach to the different setting of shared variable concurrency, building on the idea of an *interference domain* that capture relations between globally mutable variables. Like our approach it may need to reanalyze each thread repeatedly. In previous work we developed LVREs, including an ordering algorithm and a widening operator [Midtgaard et al., 2016b] and illustrated the domain with an intra-process analysis over LVRE futures. In a follow-up paper [Midtgaard et al., 2016a] we refined this idea to an inter-process analysis with LVREs for both histories and futures, albeit *limited to two synchronous processes*. The current paper generalizes from 2 to n processes by means a shuffle operator and reads off a history with $\mathcal{H}$ in favor of computing it within a fixed-point computation. Logozzo [2004] previously suggested LVREs as an abstract domain but his formulation did not fit our purpose. For one, he defines $\mathcal{L}(\epsilon) = \emptyset$ which is algebraically controversial. For another, his structural widening operator was too sensitive to syntactic variations and did not satisfy the classical widening definition [Midtgaard et al., 2016b].

9 Conclusion and perspectives

We have presented a modular approach to analyzing processes communicating by synchronous message passing. It combines the analysis results of individual processes by a dedicated shuffle operator. The approach has been formalized and proven sound for a subset of the Go programming language. We see a number of advantages to the approach: Since each analysis iteration result is sound, one can run the analysis in the background and warn of, e.g., an unsuccessful read or write, as soon as it is discovered. It also opens for algorithmic improvements to save intra-process reanalysis when futures are unchanged. Finally the analysis cache naturally falls into separate per process caches which opens up for parallelization.

Bibliography

V. Botbol, E. Chailloux, and T. L. Gall. Static analysis of communicating processes using symbolic transducers. In *VMCAI*, volume 10145 of *LNCS*, pages 73–90. Springer, 2017.

J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

C. Colby. Analyzing the communication topology of concurrent programs. In *Proc. of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 202–213, 1995.

P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. of the Second International Symposium on Programming*, pages 106–130. Dunod, France, 1976.

P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977.

P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 269–282, 1979.

P. Cousot and R. Cousot. Semantic analysis of Communicating Sequential Processes. In *Automata, Languages and Programming, 7th Colloquium*, volume 85 of *LNCS*, pages 119–133. Springer, 1980.

B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, England, second edition, 2002.

J. Feret. Confidentiality analysis of mobile systems. In *Static Analysis, 7th International Symposium, SAS 2000*, volume 1824 of *LNCS*, pages 135–154. Springer, 2000.

E. Giachino, N. Kobayashi, and C. Laneve. Deadlock analysis of unbounded process networks. In *CONCUR 2014 - Concurrency Theory - 25th International Conference. Proceedings*, volume 8704 of *LNCS*, pages 63–77. Springer, 2014.

G. Grätzer. *General Lattice Theory*. Pure and Applied Mathematics. Academic Press, 1978.

A. Igarashi and N. Kobayashi. Type-based analysis of communication for concurrent programming languages. In *Static Analysis, 4th International Symposium, SAS '97*, volume 1302 of *LNCS*, pages 187–201. Springer, 1997.

N. Kobayashi. Type-based information flow analysis for the pi-calculus. *Acta Informatica*, 42(4-5):291–347, 2005.

N. Kobayashi. A new type system for deadlock-free processes. In *CONCUR 2006 - Concurrency Theory, 17th International Conference. Proceedings*, volume 4137 of *LNCS*, pages 233–247. Springer, 2006.

N. Kobayashi and D. Sangiorgi. A hybrid type system for lock-freedom of mobile processes. In *Computer Aided Verification, 20th International Conference, CAV 2008. Proceedings*, volume 5123 of *LNCS*, pages 80–93. Springer, 2008.

J. Lange, N. Ng, B. Toninho, and N. Yoshida. Fencing off go: liveness and safety for channel-based programming. In *Proc. of the 44th Annual ACM Symposium on Principles of Programming Languages*, pages 748–761, 2017.

T. Le Gall and B. Jeannet. Lattice automata: A representation for languages on infinite alphabets, and some applications to verification. In *Static Analysis, 14th International Symposium, SAS 2007*, volume 4634 of *LNCS*, pages 52–68. Springer, 2007.

F. Logozzo. Separate compositional analysis of class-based object-oriented languages. In *Proc. of the 10th International Conference on Algebraic Methodology and Software Technology, AMAST '04*, volume 3116 of *LNCS*, pages 334–348, 2004.

N. Mercouroff. An algorithm for analyzing communicating processes. In *Proc. of the 7th International Conference on Mathematical Foundations of Programming Semantics*, volume 598 of *LNCS*, pages 312–325. Springer, 1991.

J. Midtgaard, F. Nielson, and H. R. Nielson. Iterated process analysis over lattice-valued regular expressions. In *PPDP'16: Proc. of the 18th International Symposium on Principles and Practice of Declarative Programming*, pages 132–145, 2016a.

J. Midtgaard, F. Nielson, and H. R. Nielson. A parametric abstract domain for lattice-valued regular expressions. In *Static Analysis, 23rd International Symposium, SAS 2016*, volume 9837 of *LNCS*, pages 338–360. Springer, 2016b.

A. Miné. Relational thread-modular static value analysis by abstract interpretation. In *Verification, Model Checking, and Abstract Interpretation - 15th International Conference, VMCAI 2014, Proceedings*, volume 8318 of *LNCS*, pages 39–58. Springer, 2014.

N. Ng and N. Yoshida. Static deadlock detection for concurrent go by global session graph synthesis. In *Proc. of the 25th International Conference on Compiler Construction, CC 2016*, pages 174–184. ACM, 2016.

F. Nielson and H. R. Nielson. Higher-order concurrent programs with finite communication topology. In *Proc. of the 21st Annual ACM Symposium on Principles of Programming Languages*, pages 84–97, 1994.

F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.

S. Owens, J. Reppy, and A. Turon. Regular-expression derivatives re-examined. *Journal of Functional Programming*, 19(2):173–190, 2009.

J. Reppy. *Concurrent Programming in ML.* Cambridge University Press, 1999.

R. Rydhof Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In *Static Analysis, 6th International Symposium, SAS '99*, volume 1694 of *LNCS*, pages 134–148. Springer, 1999.

C. Skalka, S. Smith, and D. Van Horn. Types and trace effects of higher order programs. *Journal of Functional Programming*, 18(2):179–249, 2008.

K. Stadtmüller, M. Sulzmann, and P. Thiemann. Static trace-based deadlock analysis for synchronous mini-go. In *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Proceedings*, volume 10017 of *LNCS*, pages 116–136, 2016.

M. Sulzmann and P. Thiemann. Derivatives for regular shuffle expressions. In *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Proceedings*, volume 8977 of *LNCS*, pages 275–286. Springer, 2015.

M. Sulzmann and P. Thiemann. Forkable regular expressions. In *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Proceedings*, volume 9618 of *LNCS*, pages 194–206. Springer, 2016.

R. E. Tarjan. Fast algorithms for solving path problems. *Journal of the ACM*, 28(3): 594–614, 1981.

A. Venet. Automatic determination of communication topologies in mobile systems. In *Static Analysis, 5th International Symposium, SAS '98*, volume 1503 of *LNCS*, pages 152–167. Springer, 1998.

A Shuffling proofs

A.1 String operation || is commutative

Proof. Let w, w' be given. We proceed by simultaneous induction on the two strings.

case $w = \epsilon$**:** By def. of $\|$ we have $\epsilon \parallel w' = \{w'\} = w' \parallel \epsilon$
case $w' = \epsilon$**:** Symmetric to the above case.
case $w = c_1 w_1, w' = c_2 w_2$**:**

$$\begin{aligned}
& c_1 w_1 \parallel c_2 w_2 \\
&= \{c_1 w \mid w \in w_1 \parallel c_2 w_2\} \;\cup\; \{c_2 w \mid w \in c_1 w_1 \parallel w_2\} && \text{(by def. of } \|\text{)} \\
&= \{c_2 w \mid w \in c_1 w_1 \parallel w_2\} \;\cup\; \{c_1 w \mid w \in w_1 \parallel c_2 w_2\} && \text{(by comm. of } \cup\text{)} \\
&= \{c_2 w \mid w \in w_2 \parallel c_1 w_1\} \;\cup\; \{c_1 w \mid w \in c_2 w_2 \parallel w_1\} && \text{(by IH)} \\
&= c_2 w_2 \parallel c_1 w_1 && \text{(by def. of } \|\text{)}
\end{aligned}$$

□

A.2 Language operation || is commutative (Lemma 2 a)

Proof. Let L_1, L_2 be given.

$$\begin{aligned}
L_1 \parallel L_2 &= \{w \mid w \in w_1 \parallel w_2 \land w_1 \in L_1 \land w_2 \in L_2\} && \text{(by def.)} \\
&= \{w \mid w \in w_2 \parallel w_1 \land w_2 \in L_2 \land w_1 \in L_1\} && \text{(by string-level comm.)} \\
&= L_2 \parallel L_1 && \text{(by def.)}
\end{aligned}$$

□

A.3 Shuffling distributes over union (Lemma 2 b)

Proof. Let L, L_1, L_2 be given.

$$\begin{aligned}
L \parallel (L_1 \cup L_2) &= \{w \mid w \in w_1 \parallel w_2 \land w_1 \in L \land w_2 \in L_1 \cup L_2\} && \text{(by def. of } \|\text{)} \\
&= \{w \mid w \in w_1 \parallel w_2 \land w_1 \in L \land w_2 \in L_1\} && \text{(by def. of } \cup\text{)} \\
&\quad \cup \{w \mid w \in w_1 \parallel w_2 \land w_1 \in L \land w_2 \in L_2\} \\
&= (L \parallel L_1) \cup (L \parallel L_2) && \text{(by def. of } \|\text{)}
\end{aligned}$$

By commutativity we immediately get $(L_1 \cup L_2) \parallel L = (L_1 \parallel L) \cup (L_2 \parallel L)$. □

A.4 Shuffling is associative (Lemma 2 c)

Proof. By definition, given a string $w \in L_1 \parallel (L_2 \parallel L_3)$ there must exist strings $w_1 \in L_1, w_2 \in L_2, w_3 \in L_3$ such that $w \in w_1 \parallel w_{23}$ and $w_{23} \in w_2 \parallel w_3$. We need to argue that similar strings exist for the right-hand-side in order for the left-hand-side $L_1 \parallel (L_2 \parallel L_3)$ to be included in the right-hand-side. We prove the property $\{w_1\} \parallel (w_2 \parallel w_3) = (w_1 \parallel w_2) \parallel \{w_3\}$ where the innermost shuffle operation is over individual strings whereas the outermost shuffle operation is over languages. This property generalizes the above to hold for any $w_{23} \in w_2 \parallel w_3$ and thereby proves the desired. We prove the property by simultaneous induction on w_1, w_2, and w_3.

base case $w_1 = \epsilon$:

$$\begin{aligned}
\{\epsilon\} \parallel (w_2 \parallel w_3) &= w_2 \parallel w_3 && \text{(by def. of lang-level }\parallel\text{)}\\
&= \{w_2\} \parallel \{w_3\} && \text{(by def. of lang-level }\parallel\text{)}\\
&= (\epsilon \parallel w_2) \parallel \{w_3\} && \text{(by def. of string-level }\parallel\text{)}
\end{aligned}$$

base case $w_2 = \epsilon$:

$$\begin{aligned}
\{w_1\} \parallel (\epsilon \parallel w_3) &= \{w_1\} \parallel \{w_3\} && \text{(by def. of string-level }\parallel\text{)}\\
&= (w_1 \parallel \epsilon) \parallel \{w_3\} && \text{(by def. of string-level }\parallel\text{)}
\end{aligned}$$

base case $w_3 = \epsilon$:

$$\begin{aligned}
\{w_1\} \parallel (w_2 \parallel \epsilon) &= \{w_1\} \parallel \{w_2\} && \text{(by def. of string-level }\parallel\text{)}\\
&= w_1 \parallel w_2 && \text{(by def. of lang-level }\parallel\text{)}\\
&= (w_1 \parallel w_2) \parallel \{\epsilon\} && \text{(by def. of lang-level }\parallel\text{)}
\end{aligned}$$

inductive step $w_1, w_2, w_3 \neq \epsilon$:

$$\begin{aligned}
&\{c_1w_1\} \parallel (c_2w_2 \parallel c_3w_3)\\
&= \{c_1w_1\} \parallel (c_2 \cdot (w_2 \parallel c_3w_3) \,\cup\, c_3 \cdot (c_2w_2 \parallel w_3)) \quad \text{(by def. of string-level }\parallel\text{)}\\
&= \{c_1w_1\} \parallel c_2 \cdot (w_2 \parallel c_3w_3) \,\cup\, \{c_1w_1\} \parallel c_3 \cdot (c_2w_2 \parallel w_3) \qquad \text{(by dist. of }\parallel\text{)}\\
&= c_1 \cdot (\{w_1\} \parallel c_2 \cdot (w_2 \parallel c_3w_3)) \,\cup\, c_2 \cdot (\{c_1w_1\} \parallel (w_2 \parallel c_3w_3))\\
&\quad \cup c_1 \cdot (\{w_1\} \parallel c_3 \cdot (c_2w_2 \parallel w_3)) \,\cup\, c_3 \cdot (\{c_1w_1\} \parallel (c_2w_2 \parallel w_3))\\
&\hfill \text{(by def. of lang-level }\parallel\text{)}\\
&= c_1 \cdot (\{w_1\} \parallel c_2 \cdot (w_2 \parallel c_3w_3) \,\cup\, \{w_1\} \parallel c_3 \cdot (c_2w_2 \parallel w_3))\\
&\quad \cup c_2 \cdot (\{c_1w_1\} \parallel (w_2 \parallel c_3w_3)) \,\cup\, c_3 \cdot (\{c_1w_1\} \parallel (c_2w_2 \parallel w_3))\\
&\hfill \text{(by dist. of }\cdot\text{, assoc. of }\cup\text{)}\\
&= c_1 \cdot (\{w_1\} \parallel (c_2 \cdot (w_2 \parallel c_3w_3) \,\cup\, c_3 \cdot (c_2w_2 \parallel w_3)))\\
&\quad \cup c_2 \cdot (\{c_1w_1\} \parallel (w_2 \parallel c_3w_3)) \,\cup\, c_3 \cdot (\{c_1w_1\} \parallel (c_2w_2 \parallel w_3)) \;\text{(by dist. of }\parallel\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= c_1 \cdot (\{w_1\} \parallel (c_2w_2 \parallel c_3w_3)) \\
&\quad \cup c_2 \cdot (\{c_1w_1\} \parallel (w_2 \parallel c_3w_3)) \ \cup\ c_3 \cdot (\{c_1w_1\} \parallel (c_2w_2 \parallel w_3)) \\
&\hspace{20em} \text{(by def. of string-level } \parallel) \\
&= c_1 \cdot ((w_1 \parallel c_2w_2) \parallel \{c_3w_3\}) \\
&\quad \cup c_2 \cdot ((c_1w_1 \parallel w_2) \parallel \{c_3w_3\}) \ \cup\ c_3 \cdot ((c_1w_1 \parallel c_2w_2) \parallel \{w_3\}) \qquad \text{(by IH)} \\
&= c_1 \cdot ((w_1 \parallel c_2w_2) \parallel \{c_3w_3\}) \ \cup\ c_2 \cdot ((c_1w_1 \parallel w_2) \parallel \{c_3w_3\}) \\
&\quad \cup c_3 \cdot ((c_1 \cdot (w_1 \parallel c_2w_2) \ \cup\ c_2 \cdot (c_1w_1 \parallel w_2)) \parallel \{w_3\}) \\
&\hspace{20em} \text{(by def. of string-level } \parallel) \\
&= c_1 \cdot ((w_1 \parallel c_2w_2) \parallel \{c_3w_3\}) \ \cup\ c_2 \cdot ((c_1w_1 \parallel w_2) \parallel \{c_3w_3\}) \\
&\quad \cup c_3 \cdot (c_1 \cdot (w_1 \parallel c_2w_2) \parallel \{w_3\} \ \cup\ c_2 \cdot (c_1w_1 \parallel w_2) \parallel \{w_3\}) \quad \text{(by dist. of } \parallel) \\
&= c_1 \cdot ((w_1 \parallel c_2w_2) \parallel \{c_3w_3\}) \ \cup\ c_2 \cdot ((c_1w_1 \parallel w_2) \parallel \{c_3w_3\}) \\
&\quad \cup c_3 \cdot (c_1 \cdot (w_1 \parallel c_2w_2) \parallel \{w_3\}) \ \cup\ c_3 \cdot (c_2 \cdot (c_1w_1 \parallel w_2) \parallel \{w_3\}) \\
&\hspace{20em} \text{(by dist. of } \cdot) \\
&= c_1 \cdot ((w_1 \parallel c_2w_2) \parallel \{c_3w_3\}) \ \cup\ c_3 \cdot (c_1 \cdot (w_1 \parallel c_2w_2) \parallel \{w_3\}) \\
&\quad \cup c_2 \cdot ((c_1w_1 \parallel w_2) \parallel \{c_3w_3\}) \ \cup\ c_3 \cdot (c_2 \cdot (c_1w_1 \parallel w_2) \parallel \{w_3\}) \\
&\hspace{20em} \text{(by assoc. of } \cup) \\
&= c_1 \cdot (w_1 \parallel c_2w_2) \parallel \{c_3w_3\} \ \cup\ c_2 \cdot (c_1w_1 \parallel w_2) \parallel \{c_3w_3\} \\
&\hspace{20em} \text{(by def. of lang-level } \parallel) \\
&= (c_1 \cdot (w_1 \parallel c_2w_2) \ \cup\ c_2 \cdot (c_1w_1 \parallel w_2)) \parallel \{c_3w_3\} \qquad \text{(by dist. of } \parallel) \\
&= (c_1w_1 \parallel c_2w_2) \parallel \{c_3w_3\} \qquad \text{(by def. of string-level } \parallel)
\end{aligned}
$$

The proof for the other direction follows symmetrically. □

A.5 Generalized shuffle property

Proof. We proceed by (nested) induction on w_1 (and w_2). Let w_1, w_2, w_3, w_4 be given.

case $w_1 = \epsilon$**:** $(\epsilon \parallel w_2) \cdot (w_3 \parallel w_4) = w_2 \cdot (w_3 \parallel w_4)$

We proceed to show $w_2 \cdot (w_3 \parallel w_4) \subseteq w_3 \parallel (w_2 \cdot w_4)$ by inner induction on w_2.

case $w_2 = \epsilon$**:** Since ϵ is the identity element for $\cdot$ we get:

$$\epsilon \cdot (w_3 \parallel w_4) = (w_3 \parallel w_4) = w_3 \parallel (\epsilon \cdot w_4)$$

case $w_2 = c_2w_2'$**:**

$$
\begin{aligned}
&c_2w_2' \cdot (w_3 \parallel w_4) \\
&= c_2 \cdot (w_2' \cdot (w_3 \parallel w_4)) && \text{(by assoc. of } \cdot) \\
&\subseteq c_2 \cdot (w_3 \parallel w_2' \cdot w_4) && \text{(by the inner IH)} \\
&\subseteq w_3 \parallel c_2 \cdot (w_2' \cdot w_4) && \text{(by the inner IH)} \\
&= w_3 \parallel (c_2 \cdot w_2') \cdot w_4 && \text{(by assoc. of } \cdot)
\end{aligned}
$$

This concludes the inner induction.

case $w_1 = c_1 w_1'$: We proceed by inner induction on w_2.

case $w_2 = \epsilon$:

$$\begin{aligned}
&(c_1 w_1' \parallel \epsilon) \cdot (w_3 \parallel w_4) \\
&= c_1 w_1' \cdot (w_3 \parallel w_4) && \text{(by def. of } \parallel) \\
&= c_1 \cdot (w_1' \parallel \epsilon) \cdot (w_3 \parallel w_4) && \text{(by def. of } \parallel) \\
&\subseteq c_1 \cdot (w_1' \cdot w_3 \parallel \epsilon \cdot w_4) && \text{(by the outer IH)} \\
&= c_1 \cdot (w_1' \cdot w_3 \parallel w_4) && (\epsilon \text{ id. for } \cdot) \\
&= (c_1 \parallel \epsilon) \cdot (w_1' \cdot w_3 \parallel w_4) && \text{(by def. of } \parallel) \\
&\subseteq (c_1 \cdot w_1' \cdot w_3) \parallel (\epsilon \cdot w_4) && \text{(by the outer IH)} \\
&= (c_1 \cdot w_1') \cdot w_3 \parallel (\epsilon \cdot w_4) && \text{(by assoc. of } \cdot)
\end{aligned}$$

case $w_2 = c_2 w_2'$:

$$\begin{aligned}
&(c_1 w_1' \parallel c_2 w_2') \cdot (w_3 \parallel w_4) \\
&= (c_1 \cdot (w_1' \parallel c_2 w_2') \,\cup\, c_2 \cdot (c_1 w_1' \parallel w_2')) \cdot (w_3 \parallel w_4) && \text{(by def. of } \parallel) \\
&= c_1 \cdot (w_1' \parallel c_2 w_2') \cdot (w_3 \parallel w_4) \,\cup\, c_2 \cdot (c_1 w_1' \parallel w_2') \cdot (w_3 \parallel w_4) \\
& && \text{(by dist. of } \cdot) \\
&\subseteq c_1 \cdot (w_1' \cdot w_3 \parallel c_2 w_2' \cdot w_4) \,\cup\, c_2 \cdot (c_1 w_1' \parallel w_2') \cdot (w_3 \parallel w_4) \\
& && \text{(by the outer IH)} \\
&\subseteq c_1 \cdot (w_1' \cdot w_3 \parallel c_2 w_2' \cdot w_4) \,\cup\, c_2 \cdot (c_1 w_1' \cdot w_3 \parallel w_2' \cdot w_4) \\
& && \text{(by the inner IH)} \\
&= (c_1 w_1' \cdot w_3) \parallel (c_2 w_2' \cdot w_4) && \text{(by def. of } \parallel)
\end{aligned}$$

$\square$

A.6 Brzozowski's equation for LVREs with shuffle

Proof. Brzozowski's equation for sub-expressions implies Brzozowski's equation for shuffle expressions:

$$
\begin{aligned}
&\mathcal{L}(r_1 \parallel r_2)\\
&= \mathcal{L}(r_1) \parallel \mathcal{L}(r_2) && \text{(by def. of } \mathcal{L}\text{)}\\
&= \mathcal{L}\Big(\sum_{a\in Atoms(A)} a(\widehat{\mathcal{D}}_a(r_1)) + \delta(r_1)\Big) \Bigg\| \mathcal{L}\Big(\sum_{a\in Atoms(A)} a(\widehat{\mathcal{D}}_a(r_2)) + \delta(r_2)\Big)\\
& && \text{(by Brzozowski's equation)}\\
&= \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a(\widehat{\mathcal{D}}_a(r_1)) + \delta(r_1))\right) \Bigg\| \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a(\widehat{\mathcal{D}}_a(r_2)) + \delta(r_2))\right)\\
& && \text{(by def. of } \mathcal{L}\text{)}\\
&= \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a)\mathcal{L}(\widehat{\mathcal{D}}_a(r_1)) \cup \mathcal{L}(\delta(r_1))\right) \Bigg\| \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a)\mathcal{L}(\widehat{\mathcal{D}}_a(r_2)) \cup \mathcal{L}(\delta(r_2))\right)\\
& && \text{(by def. of } \mathcal{L}\text{)}\\
&= \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a)\mathcal{L}(\widehat{\mathcal{D}}_a(r_1)) \cup \mathcal{L}(\delta(r_1))\right) \cdot \mathcal{L}(\delta(r_2))\\
&\quad \cup \mathcal{L}(\delta(r_1)) \cdot \left(\bigcup_{a\in Atoms(A)} \mathcal{L}(a)\mathcal{L}(\widehat{\mathcal{D}}_a(r_2)) \cup \mathcal{L}(\delta(r_2))\right)\\
&\quad \cup \bigcup_{a\in Atoms(A)} \mathcal{L}(a) \left(\mathcal{L}(\widehat{\mathcal{D}}_a(r_1)) \Bigg\| \bigcup_{a'\in Atoms(A)} \mathcal{L}(a')\mathcal{L}(\widehat{\mathcal{D}}_{a'}(r_2))\right)\\
&\quad \cup \bigcup_{a\in Atoms(A)} \mathcal{L}(a) \left(\bigcup_{a'\in Atoms(A)} \mathcal{L}(a')\mathcal{L}(\widehat{\mathcal{D}}_{a'}(r_1))\right) \Bigg\| \mathcal{L}(\widehat{\mathcal{D}}_a(r_2))\\
& && \text{(by def. of } \parallel\text{)}\\
&= \mathcal{L}(\delta(r_1)) \cdot \mathcal{L}(\delta(r_2))\\
&\quad \cup \bigcup_{a\in Atoms(A)} \mathcal{L}(a) \left(\mathcal{L}(\widehat{\mathcal{D}}_a(r_1)) \Bigg\| \bigcup_{a'\in Atoms(A)} \mathcal{L}(a')\mathcal{L}(\widehat{\mathcal{D}}_{a'}(r_2)) \cup \mathcal{L}(\delta(r_2))\right)\\
&\quad \cup \bigcup_{a\in Atoms(A)} \mathcal{L}(a) \left(\bigcup_{a'\in Atoms(A)} \mathcal{L}(a')\mathcal{L}(\widehat{\mathcal{D}}_{a'}(r_1)) \cup \mathcal{L}(\delta(r_1))\right) \Bigg\| \mathcal{L}(\widehat{\mathcal{D}}_a(r_2))\\
& && \text{(by def. of } \delta, \parallel\text{)}
\end{aligned}
$$

$$
\begin{aligned}
&= \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \Big(\mathcal{L}(\widehat{\mathcal{D}}_a(r_1)) \;\|\; \mathcal{L}(r_2) \Big) \\
&\quad \cup \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \Big(\mathcal{L}(r_1) \;\|\; \mathcal{L}(\widehat{\mathcal{D}}_a(r_2)) \Big) \\
&\quad \cup\ \mathcal{L}(\delta(r_1)) \cdot \mathcal{L}(\delta(r_2)) && \text{(by the IH)} \\
&= \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot \mathcal{L}(\widehat{\mathcal{D}}_a(r_1) \,\|\, r_2) \\
&\quad \cup \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot \mathcal{L}(r_1 \,\|\, \widehat{\mathcal{D}}_a(r_2)) \\
&\quad \cup\ \mathcal{L}(\delta(r_1 \,\|\, r_2)) && \text{(by def. of } \mathcal{L}) \\
&= \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot (\mathcal{L}(\widehat{\mathcal{D}}_a(r_1) \,\|\, r_2) \ \cup\ \mathcal{L}(r_1 \,\|\, \widehat{\mathcal{D}}_a(r_2)) \ \cup\ \mathcal{L}(\delta(r_1 \,\|\, r_2)) \\
& && \text{(by dist. of } \cdot) \\
&= \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot (\mathcal{L}(\widehat{\mathcal{D}}_a(r_1) \,\|\, r_2 \ + \ r_1 \,\|\, \widehat{\mathcal{D}}_a(r_2))) \ \cup\ \mathcal{L}(\delta(r_1 \,\|\, r_2)) \\
& && \text{(by def. of } \mathcal{L}) \\
&= \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot \mathcal{L}(\widehat{\mathcal{D}}_a(r_1 \,\|\, r_2)) \ \cup\ \mathcal{L}(\delta(r_1 \,\|\, r_2)) && \text{(by def. of } \widehat{\mathcal{D}}) \\
&= \mathcal{L}(\sum_{a \in Atoms(A)} a \cdot \widehat{\mathcal{D}}_a(r_1 \,\|\, r_2) \ + \ \delta(r_1 \,\|\, r_2)) && \text{(by def. of } \mathcal{L})
\end{aligned}
$$

□

A.7 Correctness of *nullable*

Proof. Assuming correctness for the sub-expressions we can prove correctness for a shuffle expression:

$$
\begin{aligned}
& \epsilon \in \mathcal{L}(r_1 \parallel r_2) \\
\iff & \epsilon \in \mathcal{L}(\sum_{a \in Atoms(A)} a\, \widehat{\mathcal{D}}_a(r_1 \parallel r_2) + \delta(r_1 \parallel r_2)) && \text{(by Brzozowski's equation)} \\
\iff & \epsilon \in \bigcup_{a \in Atoms(A)} \mathcal{L}(a) \cdot \mathcal{L}(\widehat{\mathcal{D}}_a(r_1 \parallel r_2)) \cup \mathcal{L}(\delta(r_1 \parallel r_2)) && \text{(by def. of } \mathcal{L}) \\
\iff & \epsilon \in \mathcal{L}(\delta(r_1 \parallel r_2)) && \text{(by def. of } \mathcal{L}, \cdot) \\
\iff & \epsilon \in \mathcal{L}(\delta(r_1)) \ \wedge \ \epsilon \in \mathcal{L}(\delta(r_2)) && \text{(by def. of } \|, \mathcal{L}) \\
\iff & nullable(r_1) \ \wedge \ nullable(r_2) && \text{(by corr. for sub.expr.)} \\
\iff & nullable(r_1 \,\|\, r_2) && \text{(by def. of } nullable)
\end{aligned}
$$

□

A.8 Finitely many derivatives

We argue that for all r, there exists at most d_r different derivatives up to ACI of $+$. We first seek a syntactic characterization of derivatives. As a warm-up Consider $\widehat{\mathcal{D}}_{a_1 a_2}(r_1 \parallel r_2)$:

$$\begin{aligned}
&\widehat{\mathcal{D}}_{a_1 a_2}(r_1 \parallel r_2) \\
&= \widehat{\mathcal{D}}_{a_2}(\widehat{\mathcal{D}}_{a_1}(r_1 \parallel r_2)) && \text{(by def. } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{a_2}(\widehat{\mathcal{D}}_{a_1}(r_1) \parallel r_2 + r_1 \parallel \widehat{\mathcal{D}}_{a_1}(r_2)) && \text{(by def. } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{a_2}(\widehat{\mathcal{D}}_{a_1}(r_1) \parallel r_2) + \widehat{\mathcal{D}}_{a_2}(r_1 \parallel \widehat{\mathcal{D}}_{a_1}(r_2)) && \text{(by def. } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{a_2}(\widehat{\mathcal{D}}_{a_1}(r_1)) \parallel r_2 + \widehat{\mathcal{D}}_{a_1}(r_1) \parallel \widehat{\mathcal{D}}_{a_2}(r_2) \\
&\qquad + \widehat{\mathcal{D}}_{a_2}(r_1) \parallel \widehat{\mathcal{D}}_{a_1}(r_2) + r_1 \parallel \widehat{\mathcal{D}}_{a_2}(\widehat{\mathcal{D}}_{a_1}(r_2)) && \text{(by def. } \widehat{\mathcal{D}}\text{)}
\end{aligned}$$

Proof. In order to help with the syntactic characterization of derivatives we introduce the short-hand notation $r_1\,[\,+\,r_2\,]^b$ with the following meaning:

$$r_1\,[\,+\,r_2\,]^b = \begin{cases} r_1 & b = 0 \\ r_1 + r_2 & b = 1 \end{cases}$$

By the definition a derivative of $r_1 \parallel r_2$ with respect to a single atom a may have up to 2 different terms. As illustrated by our example above, for a derivative $\widehat{\mathcal{D}}_{a_1 \ldots a_n}(r_1 \parallel r_2)$ there may be up to 2^n different elements in such a sum. We now prove the following syntactic characterization:

$$\begin{aligned}
&\forall r_1, r_2 \in \widehat{R}_A,\ s \in \mathit{Atoms}(A)^*. \\
&\quad \exists b_1, \ldots, b_m \in \{0,1\},\ s_1, \ldots, s_m,\ s'_1, \ldots, s'_m \in \mathit{Atoms}(A)^*. \\
&\quad\quad \widehat{\mathcal{D}}_s(r_1 \parallel r_2) = \sum_{1 \le i \le m} [\,\widehat{\mathcal{D}}_{s_i}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i}(r_2)\,]^{b_i} \ \text{ where } m = 2^{|s|}
\end{aligned}$$

We proceed by induction on s:

case $s = \epsilon$**:** $|\epsilon| = 0$ hence there exists $s_1 = \epsilon$, $s'_1 = \epsilon$, and $b_1 = 1$ such that

$$\widehat{\mathcal{D}}_\epsilon(r_1 \parallel r_2) = r_1 \parallel r_2 = \sum_{1 \le i \le 2^0} [\,\widehat{\mathcal{D}}_{s_i}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i}(r_2)\,]^{b_i}$$

case $s = s'a$**:**

$$\begin{aligned}
&\widehat{\mathcal{D}}_{s'a}(r_1 \parallel r_2) \\
&= \widehat{\mathcal{D}}_a(\widehat{\mathcal{D}}_{s'}(r_1 \parallel r_2)) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_a\Big(\sum_{1 \le i \le m'} [\,\widehat{\mathcal{D}}_{s_i}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i}(r_2)\,]^{b_i}\ \Big) && \text{(by IH)} \\
&= \sum_{1 \le i \le m'} [\,\widehat{\mathcal{D}}_a(\widehat{\mathcal{D}}_{s_i}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i}(r_2))\,]^{b_i} && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \sum_{1 \le i \le m'} [\,\widehat{\mathcal{D}}_{s_i a}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i}(r_2)\,]^{b_i} + [\,\widehat{\mathcal{D}}_{s_i}(r_1) \parallel \widehat{\mathcal{D}}_{s'_i a}(r_2)\,]^{b_i} && \text{(by def. of } \widehat{\mathcal{D}}\text{)}
\end{aligned}$$

for $m' = 2^{|s'|}$ values of $b_1, \ldots, b_{m'}$, $s_1, \ldots, s_{m'}$, and $s'_1, \ldots, s'_{m'}$. In terms arising from the first line we add the atom a to either the sequence s_i or s'_i to form corresponding sequences for the inductively formed term while preserving the corresponding value of b_i. We can do so with $2 \cdot m'$ Boolean and sequence values b_j, s_j, s'_j. We thereby need $2 \cdot m' = 2 \cdot 2^{|s'|} = 2^{|s'|+1} = 2^{|s'a|}$ different Boolean and sequence values b_j, s_j, s'_j.

There are only as many different derivatives (up to ACI of $+$) as there are different sets of such pairs. For each of the d_{r_1} different first components in such pairs there are at most d_{r_2} different second components and hence at most $d_{r_1} * d_{r_2}$ different pairs. This gives an upper bound of $2^{d_{r_1} * d_{r_2}}$ different sets of such pairs. □

A.9 $\widehat{range}$ partitions (Lemma 8)

Proof. Let r_1, r_2, $[a_i] \in \widehat{range}(r_1 \parallel r_2)$, $a, a' \in Atoms(A)$ be given and assume that $a, a' \in [a_i]$. By the IH we can furthermore assume the property for r_1 and r_2. Since $a, a' \in [a_i] \in \widehat{range}(r_1 \parallel r_2) = \widehat{overlay}(\widehat{range}(r_1), \widehat{range}(r_2))$ we must have $a, a' \in [a_1] \in \widehat{range}(r_1)$ and $a, a' \in [a_2] \in \widehat{range}(r_2)$ for some equivalence classes $[a_1]$ and $[a_2]$ since $\widehat{overlay}$ computes a partition refinement of both $\widehat{range}(r_1)$ and $\widehat{range}(r_2)$. We therefore have

$$\begin{aligned}
\widehat{\mathcal{D}}_a(r_1 \parallel r_2) &= \widehat{\mathcal{D}}_a(r_1) \parallel r_2 \ + \ r_1 \parallel \widehat{\mathcal{D}}_a(r_2) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{a'}(r_1) \parallel r_2 \ + \ r_1 \parallel \widehat{\mathcal{D}}_{a'}(r_2) && \text{(by the above, IH)} \\
&= \widehat{\mathcal{D}}_{a'}(r_1 \parallel r_2) && \text{(by def. of } \widehat{\mathcal{D}}\text{)}
\end{aligned}$$

□

B Soundness proofs

B.1 One step statement soundness, terminal (Lemma 10)

We first observe that for $\alpha = \tau$ we have

$$\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{D}}_{\widehat{\epsilon}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{D}}_{\epsilon}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s))$$

which we utilize in the proof of both Lemmas 11 and 10.

Proof. Let s, ρ, ρ_1, α, $\widehat{\mathcal{E}^i}$, $\widehat{\mathcal{X}^i}$ be given. Assume $\langle s, \rho\rangle \xrightarrow{\alpha} \rho_1$, $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s$, and $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s)))$.

case SKIP**:** By assumption $\langle \mathtt{skip}^\ell, \rho\rangle \xrightarrow{\tau} \rho$, $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \mathtt{skip}^\ell$, and $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\ell))$. But then $\mathit{last}(\mathtt{skip}^\ell) = \{\ell\}$ and $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\ell)) \subseteq \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell))$ by the analysis specification and monotonicity of γ_{st}. Furthermore $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s)) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell)$.

case ASSIGN**:** By assumption $\langle x =^\ell e, \rho\rangle \xrightarrow{\tau} \rho[x \mapsto v]$ where $\rho \vdash_{\mathcal{A}} e \Downarrow v$ and $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models x =^\ell e$ and $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\ell))$. But then $\mathit{last}(x =^\ell e) = \{\ell\}$, $v \in \gamma_v(\widehat{\mathcal{A}}(e, \widehat{\mathcal{E}^i_\rho}(\ell)))$, and hence $\rho[x \mapsto v] \in \gamma_{st}(\widehat{\mathit{assign}}(\widehat{\mathcal{E}^i_\rho}(\ell), x, \widehat{\mathcal{A}}(e, \widehat{\mathcal{E}^i_\rho}(\ell)))) \subseteq \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell))$ by Lemma 9 and the analysis specification. Again $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s)) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell)$ follows by the analysis specification.

case FOR2**:** By assumption we have $\langle \mathtt{for}\ b^\ell\ \{\ s_1\ \}, \rho\rangle \xrightarrow{\tau} \rho$ and $\rho \vdash_{\mathcal{B}} b \Downarrow \mathtt{ff}$ from the semantics and $(\widehat{\mathit{false}}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{X}^i}(\ell)$ where $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$. From Lemma 9 and monotonicity of γ_{st} we therefore get $\rho \in \gamma_{st}(\widehat{\mathit{false}}(b, \widehat{\mathcal{E}^i_\rho}(\ell))) \subseteq \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell))$. From the above we have $\widehat{\mathcal{E}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell)$ hence $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s)) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell)$.

The cases SEQ1, SEQ2, IF1, IF2, SELECT, READ, WRITE, and FOR1 are vacuously true as they do not lead to a terminal configuration.

B.2 One step statement soundness, non-terminal (Lemma 11)

We warm up with a helper lemma:

Lemma 16 (Preservation of *last***).** $\langle s, \rho\rangle \xrightarrow{\alpha} \langle s', \rho'\rangle \implies \mathit{last}(s') \subseteq \mathit{last}(s)$

Proof. By structural induction on s. Let s, s_1, ρ, ρ_1, α, $\widehat{\mathcal{E}^i}$, $\widehat{\mathcal{X}^i}$ be given. Assume $\langle s, \rho\rangle \xrightarrow{\alpha} \langle s_1, \rho_1\rangle$, $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s$, $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s)))$, and $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s))) \not\sqsubseteq_{\approx} \emptyset$.

case SEQ1**:** Then $\langle s_1\ \mathtt{;}\ s_2, \rho\rangle \xrightarrow{\alpha} \langle s_3\ \mathtt{;}\ s_2, \rho_1\rangle$ and $\langle s_1, \rho\rangle \xrightarrow{\alpha} \langle s_3, \rho_1\rangle$. Furthermore $\mathit{first}(s_1\ \mathtt{;}\ s_2) = \mathit{first}(s_1)$ and from the analysis specification we therefore have $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1$. Since $\rho \in \widehat{\mathcal{E}^i}(\mathit{first}(s_1\ \mathtt{;}\ s_2)) = \widehat{\mathcal{E}^i}(\mathit{first}(s_1))$ we can therefore apply the induction hypothesis and conclude $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_3$, $\rho_1 \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_3)))$ and $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_3))$. Furthermore $\mathit{first}(s_3) = \mathit{first}(s_3\ \mathtt{;}\ s_2)$.

(Part 1): We need to show $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_3\ \mathtt{;}\ s_2$ or equivalently (a) $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_3$, (b) $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_2$, and (c) $\widehat{\mathcal{X}^i}(\ell_3) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_2))$ for all $\ell_3 \in \mathit{last}(s_3)$.

(a) follows immediately from the induction hypothesis and (b) follows from the assumption $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1\ \mathtt{;}\ s_2$. From the same assumption we furthermore have $\widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_2))$ for all $\ell_1 \in \mathit{last}(s_1)$ which together with $\mathit{last}(s_3) \subseteq \mathit{last}(s_1)$ from Lemma 16 means that $\widehat{\mathcal{X}^i}(\ell_3) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_2))$ for all $\ell_3 \in \mathit{last}(s_3) \subseteq \mathit{last}(s_1)$ and thus yields (c).

For parts 2 and 3 since $\widehat{\mathcal{E}^i}(\mathit{first}(s_3)) = \widehat{\mathcal{E}^i}(\mathit{first}(s_3\ \mathtt{;}\ s_2))$ we therefore have $\rho_1 \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_3))) = \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_3\ \mathtt{;}\ s_2)))$ and also $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_1\ \mathtt{;}\ s_2))) = \widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_3)) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s_3\ \mathtt{;}\ s_2))$.

case SEQ2**:** Then $\langle s_1\ \mathtt{;}\ s_2, \rho\rangle \xrightarrow{\alpha} \langle s_2, \rho_1\rangle$ and $\langle s_1, \rho\rangle \xrightarrow{\alpha} \rho_1$. Furthermore we have $\mathit{first}(s_1\ \mathtt{;}\ s_2) = \mathit{first}(s_1)$ and from the analysis specification we therefore have $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1$, $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_2$, and $\forall \ell \in \mathit{last}(s_1).\ \widehat{\mathcal{X}^i}(\ell) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_2))$. Since $\rho \in \widehat{\mathcal{E}^i}(\mathit{first}(s_1\ \mathtt{;}\ s_2)) = \widehat{\mathcal{E}^i}(\mathit{first}(s_1))$ we can therefore apply Lemma 10 and conclude $\rho_1 \in \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell))$ and $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell)$ for all $\ell \in \mathit{last}(s_1)$. As a consequence for any $\ell \in \mathit{last}(s_1)$ we have $\rho_1 \in \gamma_{st}(\widehat{\mathcal{X}^i_\rho}(\ell)) \subseteq \gamma_{st}(\widehat{\mathcal{E}^i}(\mathit{first}(s_2)))$ (by monotonicity of γ_{st}) and $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))) \sqsubseteq_{\approx} \widehat{\mathcal{X}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_2))$.

case IF1**:** Then $\langle \mathtt{if}\ b^\ell\ \{\, s_1 \,\}\ \mathtt{else}\ \{\, s_2 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_1, \rho\rangle$ and $\rho \vdash_{\mathcal{B}} b \Downarrow \mathtt{tt}$. Furthermore $\mathit{first}(\mathtt{if}\ b^\ell\ \{\, s_1 \,\}\ \mathtt{else}\ \{\, s_2 \,\}) = \ell$ and from the analysis specification we have that $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1$ and $(\widehat{\mathit{true}}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_1))$ where $(\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$.

Part 1 follows immediately from the analysis specification. For part 2 from Lemma 9 and monotonicity of γ_{st} we get $\rho \in \gamma_{st}(\widehat{\mathit{true}}(b, \widehat{\mathcal{E}^i_\rho}(\ell))) \subseteq \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_1)))$. For part 3 we know $\widehat{\mathcal{E}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))$ and therefore $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\ell)) = \widehat{\mathcal{E}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))$.

case IF2**:** Then $\langle \mathtt{if}\ b^\ell\ \{\, s_1 \,\}\ \mathtt{else}\ \{\, s_2 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_2, \rho\rangle$ and $\rho \vdash_{\mathcal{B}} b \Downarrow \mathtt{ff}$. Furthermore $\mathit{first}(\mathtt{if}\ b^\ell\ \{\, s_1 \,\}\ \mathtt{else}\ \{\, s_2 \,\}) = \ell$ and from the analysis specification we have that $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_2$ and $(\widehat{\mathit{false}}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_2))$ where $(\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$.

Part 1 follows immediately from the analysis specification. For part 2 from Lemma 9 and monotonicity of γ_{st} we get $\rho \in \gamma_{st}(\widehat{\mathit{false}}(b, \widehat{\mathcal{E}^i_\rho}(\ell))) \subseteq \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_2)))$. For part 3 we know $\widehat{\mathcal{E}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_2))$ and therefore $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\ell)) = \widehat{\mathcal{E}^i_f}(\ell) \sqsubseteq_{\approx} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_2))$.

case FOR1**:** By assumption we have $\langle \mathtt{for}\ b^\ell\ \{\, s_1 \,\}, \rho\rangle \xrightarrow{\tau} \langle s_1\ \mathtt{;}\ \mathtt{for}\ b^\ell\ \{\, s_1 \,\}, \rho\rangle$ and $\rho \vdash_{\mathcal{B}} b \Downarrow \mathtt{tt}$ from the semantics and $(\widehat{\mathit{true}}(b, \widehat{\rho}), \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^i}(\mathit{first}(s_1))$ where $(\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$ and for all $\ell_1 \in \mathit{last}(s_1).\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq \widehat{\mathcal{E}^i}(\ell)$ from the analysis specification. Furthermore we have $\mathit{first}(s_1\ \mathtt{;}\ \mathtt{for}\ b^\ell\ \{\, s_1 \,\}) = \mathit{first}(s_1)$.

For part 1 we need to argue that $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1\ \mathtt{;}\ \mathtt{for}\ b^\ell\ \{\, s_1 \,\}$ meaning that (a) $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1$, (b) $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \mathtt{for}\ b^\ell\ \{\, s_1 \,\}$, and (c) $\forall \ell_1 \in \mathit{last}(s_1).\ \widehat{\mathcal{X}^i}(\ell_1) \sqsubseteq$

$\widehat{\mathcal{E}}^i(\mathit{first}(s_1))$. This is however immediate as (b) (and consequently (a)) and (c) all follow from our assumptions.
Part 2 follows from Lemma 9 and monotonicity of γ_{st}: $\rho \in \gamma_{st}(\widehat{\mathit{true}}(b, \widehat{\mathcal{E}}^i_\rho(\ell))) \subseteq \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_1))) = \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_1 \texttt{ ; for } b^\ell \texttt{ \{ } s_1 \texttt{ \}})))$. For part 3 we have $\widehat{\mathcal{E}^i_f}(\ell) \mathrel{\widehat{\sqsubseteq}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_1))$ hence $\widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\ell)) = \widehat{\mathcal{E}^i_f}(\ell) \mathrel{\widehat{\sqsubseteq}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_1)) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s_1 \texttt{ ; for } b^\ell \texttt{ \{ } s_1 \texttt{ \}}))$.

case SELECT**:** By assumption we have $\langle \texttt{select}^\ell \texttt{ \{ } a_1 \dots a_n \texttt{ \}}, \rho\rangle \xrightarrow{\alpha} \langle s_j, \rho'\rangle$ and $\langle a_j, \rho\rangle \xrightarrow{\alpha} \langle s_j, \rho'\rangle$ for some $1 \leq j \leq n$. Furthermore we have $\widehat{\mathcal{E}}^i, \widehat{\mathcal{X}}^i \models a_j$ and $\widehat{\mathcal{E}}^i(\ell) \sqsubseteq \widehat{\mathcal{E}}^i(\mathit{first}(a_j))$ from the analysis specification and $\mathit{first}(\texttt{select}^\ell \texttt{ \{ } a_1 \dots a_n \texttt{ \}}) = \ell$. We therefore have $\rho \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\ell)) \subseteq \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(a_j)))$ by monotonicity of γ_{st} and hence by the IH (with s ranging over both statements and cases) we conclude $\widehat{\mathcal{E}}^i, \widehat{\mathcal{X}}^i \models s_j$, $\rho' \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_j)))$ and $\widehat{\mathcal{D}}_{\widehat{\alpha}}(\widehat{\mathcal{E}^i_f}(\ell)) \mathrel{\widehat{\sqsubseteq}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_j))$ as desired.

case READ**:** By assumption we have $\langle \texttt{case } x \texttt{ = <-}^\ell \mathit{ch} \texttt{ : } s, \rho\rangle \xrightarrow{\mathit{ch}?v} \langle s, \rho[x \mapsto v]\rangle$ and $\widehat{\mathcal{E}}^i, \widehat{\mathcal{X}}^i \models \texttt{case } x \texttt{ = <-}^\ell \mathit{ch} \texttt{ : } s$. By definition $\mathit{first}(\texttt{case } x \texttt{ = <-}^\ell \mathit{ch} \texttt{ : } s) = \ell$ and by assumption $\rho \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\ell))$, and $\widehat{\mathcal{D}}_{\widehat{\overline{\mathit{ch}?v}}}(\widehat{\mathcal{E}^i_f}(\ell)) \mathrel{\widehat{\not\sqsubseteq}} \emptyset$.
We have that $[\mathit{ch};\mathit{ch}]$ is an atom in $\mathit{Interval}$, that $\alpha_v(\{v\})$ is an atom in $\widehat{\mathit{Val}}$, that $([\mathit{ch};\mathit{ch}], \alpha_v(\{v\}))$ is an atom in $\widehat{\mathit{Write}}(\widehat{\mathit{Val}})$, and that

$$\begin{aligned}
\widehat{\overline{\mathit{ch}?v}} = \widehat{\mathit{ch}!v} = \alpha_{\mathit{ch}}(\{\mathit{ch}!v\}) &= (\alpha_{wr}(\{\mathit{ch}!v\}), \alpha_{rd}(\emptyset)) \\
&= (\alpha_{wr}(\{\mathit{ch}!v\}), (\bot, \bot)) \\
&= ((\alpha_{\mathit{Int}}(\{\mathit{ch}\}), \alpha_v(\{v\})), (\bot, \bot)) \\
&= (([\mathit{ch};\mathit{ch}], \alpha_v(\{v\})), (\bot, \bot)) \\
&= \mathit{ch}!\alpha_v(\{v\})
\end{aligned}$$

is an atom in $\widehat{\mathit{Ch}}(\widehat{\mathit{Val}})$. Let $(\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}}^i(\ell)$. Since $\widehat{\mathcal{D}}_{\widehat{\mathit{ch}!v}}(\widehat{f}) \mathrel{\widehat{\not\sqsubseteq}} \emptyset$ there exists an equivalence class $[\mathit{ch}!\widehat{v}_a] \in \widehat{\mathit{range}}(\widehat{f})$ such that $\alpha_{\mathit{ch}}(\{\mathit{ch}!v\}) = \mathit{ch}!(\alpha_v(\{v\})) \in [\mathit{ch}!\widehat{v}_a]$ and $\widehat{\mathcal{D}}_{\alpha_{\mathit{ch}}(\{\mathit{ch}!v\})}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([\mathit{ch}!\widehat{v}_a])}(\widehat{f}) \mathrel{\widehat{\not\sqsubseteq}} \emptyset$.
Furthermore, by our assumption about $\widehat{\mathit{project}}$: $\alpha_{\mathit{ch}}(\{\mathit{ch}!v\}) = \mathit{ch}!(\alpha_v(\{v\})) \sqsubseteq \widehat{\mathit{project}}([\mathit{ch}!\widehat{v}_a]) = \mathit{ch}!\widehat{v}$ and therefore

$$\begin{aligned}
\mathit{ch}!v &\in \gamma_{\mathit{ch}}(\widehat{\mathit{project}}([\mathit{ch}!\widehat{v}_a])) \\
&= \gamma_{\mathit{ch}}(\mathit{ch}!\widehat{v}) \\
&= \gamma_{wr}((\mathit{ch}, \widehat{v})) \\
&= \{[\mathit{ch};\mathit{ch}]!v \mid \mathit{ch} \in \gamma_{\mathit{Int}}([\mathit{ch};\mathit{ch}]) \ \land \ v \in \gamma_v(\widehat{v})\}
\end{aligned}$$

which means that $v \in \gamma_v(\widehat{v})$. But then by the implication in the analysis specification $\widehat{\mathit{assign}}(\widehat{\rho}, x, \widehat{v}) \sqsubseteq \widehat{\mathcal{X}}^i_\rho(\ell) \sqsubseteq \widehat{\mathcal{E}}^i_\rho(\mathit{first}(s))$ and $\widehat{\mathcal{D}}_{\widehat{\mathit{repr}}([\mathit{ch}!\widehat{v}_a])}(\widehat{f}) \mathrel{\widehat{\sqsubseteq}} \widehat{\mathcal{X}^i_f}(\ell) \mathrel{\widehat{\sqsubseteq}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s))$.
Part 1 $\widehat{\mathcal{E}}^i, \widehat{\mathcal{X}}^i \models s$ now follows immediately from the analysis specification.
Part 2 follows from Lemma 9 and the Galois connection properties of α_{st} and γ_{st}: $\rho[x \mapsto v] \in \gamma_{st}(\widehat{\mathit{assign}}(\widehat{\rho}, x, \widehat{v})) \subseteq \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s)))$,

Part 3 also follows from the above: $\widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a])}(\widehat{f}) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{X}^i_f}(\ell) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s))$.

case WRITE**:** By assumption we have $\langle \texttt{case}\ ch\ \texttt{<-}^{\ell}\ e\ \texttt{:}\ s, \rho\rangle \xrightarrow{ch!v} \langle s, \rho\rangle$ and $\rho \vdash_{\mathcal{A}} e \Downarrow v$ from the semantics. By definition we have $\mathit{first}(\texttt{case}\ ch\ \texttt{<-}^{\ell}\ e\ \texttt{:}\ s) = \ell$ and by assumption $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models \texttt{case}\ ch\ \texttt{<-}^{\ell}\ e\ \texttt{:}\ s$, $\rho \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\ell))$, and $\widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{E}^i_f}(\ell)) \not\mathrel{\underset{\sim}{\sqsubset}} \emptyset$. Furthermore

$$\begin{aligned}\widehat{ch!v} = \widehat{ch?v} &= \alpha_{ch}(\{ch?v\}) \\ &= (\alpha_{wr}(\emptyset), \alpha_{rd}(\{ch?v\})) \\ &= ((\bot,\bot), \alpha_{rd}(\{ch?v\})) \\ &= ((\bot,\bot), (\alpha_{\mathit{Int}}(\{ch\}), \alpha_v(\{v\}))) \\ &= ((\bot,\bot), ([ch;ch], \alpha_v(\{v\}))) \\ &= ch?\alpha_v(\{v\})\end{aligned}$$

is an atom in $\widehat{Ch}(\widehat{Val})$. Let $(\widehat{\rho}, \widehat{f}) = \widehat{\mathcal{E}^i}(\ell)$. Hence there exists an equivalence class $[ch?\widehat{v}_a] \in \widehat{range}(\widehat{f})$ such that $\alpha_{ch}(\{ch?v\}) = ch?(\alpha_v(\{v\})) \in [ch?\widehat{v}_a]$ and $\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch?v\})}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{f}) \not\mathrel{\underset{\sim}{\sqsubset}} \emptyset$.
Furthermore, by our assumption about $\widehat{project}$: $\alpha_{ch}(\{ch?v\}) = ch?(\alpha_v(\{v\})) \sqsubseteq \widehat{project}([ch?\widehat{v}_a]) = ch?\widehat{v}$ and therefore

$$\begin{aligned}ch?v \in\ &\gamma_{ch}(\widehat{project}([ch?\widehat{v}_a])) \\ &= \gamma_{ch}(ch?\widehat{v}) \\ &= \gamma_{rd}((ch, \widehat{v})) \\ &= \{[ch;ch]?v \mid ch \in \gamma_{\mathit{Int}}([ch;ch]) \ \wedge\ v \in \gamma_v(\widehat{v})\}\end{aligned}$$

which means that $v \in \gamma_v(\widehat{v})$. But by Lemma 9 we also have $v \in \gamma_v(\widehat{v}')$ for $\widehat{v}' = \widehat{\mathcal{A}}(\widehat{\rho}, e)$ hence $v \in \gamma_v(\widehat{v}) \cap \gamma_v(\widehat{v}') = \gamma_v(\widehat{v} \sqcap \widehat{v}')$ which means that $\widehat{v} \sqcap \widehat{v}' \neq \bot$. But then by the implication in the analysis specification $\widehat{\rho} \sqsubseteq \widehat{\mathcal{X}^i_\rho}(\ell) \sqsubseteq \widehat{\mathcal{E}^i_\rho}(\mathit{first}(s))$ and $\widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{f}) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{X}^i_f}(\ell) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s))$.
Part 1 $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s$ now follows immediately from the analysis specification.
Part 2 follows from our assumptions, by monotonicity of γ_{st}, and transitivity of $\subseteq$: $\rho \in \gamma_{st}(\widehat{\rho}) \subseteq \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s)))$.
Part 3 follows by the above and transitivity: $\widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{E}^i_f}(\ell)) = \widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{\mathcal{E}^i_f}(\ell)) = \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{\mathcal{E}^i_f}(\ell)) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s))$

The cases SKIP, ASSIGN, and FOR2 are vacuously true as they lead to a terminal configuration.

B.3 Analysis soundness (Theorem 12)

Proof. We prove the following generalization from which analysis soundness follows immediately as a corollary:

For all programs $s_1 : \cdots : s_n$, initial stores ρ_{init}, acceptable analysis answers $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}$ such that $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 : \cdots : s_n$, $\forall i.\ \rho_{init} \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_i)))$, and arbitrary traces $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ such that $\forall i.\ f_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$ then for any i such that $1 \le i \le n$ and $c'_i = \langle s'_i, \rho'_i\rangle$ we have

$$\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s'_i \ \wedge\ \widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_k)}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i)) \ \wedge\ \rho'_i \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s'_i)))$$

We proceed by induction on the length of the trace.

case $k = 0$**:** Then for any i we have $\langle s_i, \rho_{init}\rangle = c'_i = \langle s'_i, \rho'_i\rangle$ and $\alpha_1 \ldots \alpha_k = \epsilon$. Since $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_1 : \cdots : s_n$ we immediately have $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s_i$ and $\widehat{\mathcal{D}}_\epsilon(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) = \widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))$ by reflexivity. Furthermore $\rho_{init} \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s_i)))$ follows from our assumptions.

case $k = k' + 1$**:** Given a program, a solution to the analysis specification, and a trace $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_{k'}}{\Longrightarrow} c'_1 \ldots c'_n \overset{\alpha_{k'+1}}{\Longrightarrow} c''_1 \ldots c''_n$ satisfying the above requirements, by the induction hypothesis for any $c'_i = \langle s'_i, \rho'_i\rangle$ we have $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s'_i$, $\widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_{k'})}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))$, and $\rho'_i \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s'_i)))$. We proceed by case analysis on the applied system rule.

case SYSTAU**:** $c'_1 \ldots c'_n \overset{i,\tau}{\Longrightarrow} c''_1 \ldots c''_n$ for some process number $1 \le i \le n$. Furthermore we know that $\langle s'_i, \rho'_i\rangle \xrightarrow{\tau} \langle s''_i, \rho''_i\rangle$ and that for $k \neq i$ we have $c'_k = \langle s'_k, \rho'_k\rangle = \langle s''_k, \rho''_k\rangle = c''_k$, $\widehat{\mathcal{E}^k}, \widehat{\mathcal{X}^k} \models s''_k$, $\rho''_k = \rho'_k \in \gamma_{st}(\widehat{\mathcal{E}^k_\rho}(\mathit{first}(s'_k))) = \gamma_{st}(\widehat{\mathcal{E}^k_\rho}(\mathit{first}(s''_k)))$ and

$$\begin{aligned}
\widehat{\mathcal{D}}_{\widehat{f_k(\alpha_1 \ldots \alpha_{k'} \cdot (i,\tau))}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k))) &= \widehat{\mathcal{D}}_{\widehat{f_k(i,\tau)}}(\widehat{\mathcal{D}}_{\widehat{f_k(\alpha_1 \ldots \alpha_{k'})}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k)))) \\
&\qquad\qquad \text{(by def. of } \widehat{\mathcal{D}}) \\
&= \widehat{\mathcal{D}}_{\widehat{f_k(\alpha_1 \ldots \alpha_{k'})}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k))) \\
&\qquad\qquad \text{(by def. of } f_k, \widehat{\mathcal{D}}) \\
&\mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k)) \qquad \text{(by IH)} \\
&= \widehat{\mathcal{E}^k_f}(\mathit{first}(s''_k)) \qquad \text{(by the above)}
\end{aligned}$$

By assumption $f_i(\alpha_1 \ldots \alpha_{k'}(i,\tau)) = f_i(\alpha_1 \ldots \alpha_{k'}) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$ hence

$$\begin{aligned}
&\widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_{k'}(i,\tau))}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \\
&= \widehat{\mathcal{D}}_{\widehat{f_i(i,\tau)}}(\widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_{k'})}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))) && \text{(by def. of } \widehat{\mathcal{D}}) \\
&\mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{D}}_\epsilon(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) && \text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\
&\not\mathrel{\underset{\sim}{\sqsubset}} \emptyset && \text{(by Lemma 5)}
\end{aligned}$$

Hence by Lemma 11 we get $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \models s''_i$ and $\rho''_i \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s''_i)))$. Finally since $\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i)) = \widehat{\mathcal{D}}_\epsilon(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) = \widehat{\mathcal{D}}_{\widehat{\tau}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) \mathrel{\underset{\sim}{\sqsubset}} \widehat{\mathcal{E}^i_f}(\mathit{first}(s''_i))$

and $\widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_{k'} \cdot (i,\tau))}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \precsim \widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))$ by the above we can conclude that $\widehat{\mathcal{D}}_{\widehat{f_i(\alpha_1 \ldots \alpha_{k'} \cdot (i,\tau))}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \precsim \widehat{\mathcal{E}^i_f}(\mathit{first}(s''_i))$ by transitivity of $\precsim$.

case SYSCOMM**:** We know that $\langle s'_i, \rho'_i\rangle \xrightarrow{ch!v} \langle s''_i, \rho''_i\rangle$ and $\langle s'_j, \rho'_j\rangle \xrightarrow{ch?v} \langle s''_j, \rho''_j\rangle$ for some i, j such that $i \neq j$.

For $k \notin \{i, j\}$ we have $c'_k = \langle s'_k, \rho'_k\rangle = \langle s''_k, \rho''_k\rangle = c''_k$ and therefore we immediately get $\widehat{\mathcal{E}^k}, \widehat{\mathcal{X}^k} \models s''_k$ and $\rho''_k \in \gamma_{st}(\widehat{\mathcal{E}^k_\rho}(\mathit{first}(s''_k)))$.

By assumption $f_k(\alpha_1 \ldots \alpha_{k'}(i, ch, v, j)) = f_k(\alpha_1 \ldots \alpha_{k'}) \cdot f_k(i, ch, v, j) \in \mathcal{L}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k)))$ and by the induction hypothesis we have

$$\widehat{\mathcal{D}}_{f_k(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k))) \precsim \widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))$$

Hence by Lemma 5 we have either $ch!v \cdot ch?v \in \mathcal{L}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k)))$ (when $i < j$) or $ch?v \cdot ch!v \in \mathcal{L}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k)))$ (when $i > j$). As a consequence when $i < j$ according to the analysis specification there must exist an equivalence class $[ch'!\widehat{v}']$ such that $\widehat{ch!v} \in [ch'!\widehat{v}']$ with the property that

$$\begin{aligned}\widehat{\mathcal{D}}_{\widehat{ch!v \cdot ch?v}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))) &= \widehat{\mathcal{D}}_{\widehat{repr}([ch'!\widehat{v}']) \cdot \widehat{repr}([ch'?\widehat{v}'])}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))) \\ &\precsim \widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))\end{aligned}$$

(and similarly when $i > j$). Hence we have

$$\begin{aligned}\widehat{\mathcal{D}}_{f_k(\alpha_1 \ldots \widehat{\alpha_{k'} \cdot (i,ch,v,j)})}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k))) &= \widehat{\mathcal{D}}_{f_k(\widehat{i,ch,v,j})}(\widehat{\mathcal{D}}_{f_k(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s_k)))) \\ &\qquad \text{(by def. of } \widehat{\mathcal{D}}) \\ &\precsim \widehat{\mathcal{D}}_{f_k(\widehat{i,ch,v,j})}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))) \\ &\qquad \text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\ &= \begin{cases} \widehat{\mathcal{D}}_{\widehat{ch!v \cdot ch?v}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))) & i < j \\ \widehat{\mathcal{D}}_{\widehat{ch?v \cdot ch!v}}(\widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k))) & i > j \end{cases} \\ &\qquad \text{(by def. of } f_k) \\ &\precsim \widehat{\mathcal{E}^k_f}(\mathit{first}(s'_k)) \qquad \text{(by the above)} \\ &= \widehat{\mathcal{E}^k_f}(\mathit{first}(s''_k)) \qquad \text{(by further above)}\end{aligned}$$

from our assumptions.

i case: By assumption $f_i(\alpha_1 \ldots \alpha_{k'}(i,ch,v,j)) = f_i(\alpha_1 \ldots \alpha_{k'}) \cdot ch!v \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$ hence

$$\begin{aligned}
&\widehat{\mathcal{D}}_{f_i(\alpha_1 \ldots \widehat{\alpha_{k'}(i},ch,v,j))}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) \\
&= \widehat{\mathcal{D}}_{f_i(\widehat{i,ch},v,j)}(\widehat{\mathcal{D}}_{f_i(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{D}}_{f_i(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&\lesssim \widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) && \text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\
&\not\lesssim \emptyset && \text{(by Lemma 5)}
\end{aligned}$$

Hence from Lemma 11 we get $\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i} \vDash s''_i$ and $\rho''_i \in \gamma_{st}(\widehat{\mathcal{E}^i_\rho}(\mathit{first}(s''_i)))$. Finally

$$\begin{aligned}
\widehat{\mathcal{D}}_{f_i(\alpha_1 \ldots \widehat{\alpha_{k'} \cdot (i},ch,v,j))}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i))) &= \widehat{\mathcal{D}}_{f_i(\widehat{i,ch},v,j)}(\widehat{\mathcal{D}}_{f_i(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))) \\
&\qquad \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{\mathcal{D}}_{f_i(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))) \\
&\qquad \text{(by def. of } f_i\text{)} \\
&\lesssim \widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) \\
&\qquad \text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\
&= \widehat{\mathcal{D}}_{\overline{\widehat{ch!v}}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) \qquad \text{(by def. of } \bar{\cdot}\text{)} \\
&\lesssim \widehat{\mathcal{E}^i_f}(\mathit{first}(s''_i)) \qquad \text{(by Lemma 11)}
\end{aligned}$$

j case: By assumption $f_j(\alpha_1 \ldots \alpha_{k'}(i,ch,v,j)) = f_j(\alpha_1 \ldots \alpha_{k'}) \cdot ch?v \in \mathcal{L}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j)))$ hence

$$\begin{aligned}
&\widehat{\mathcal{D}}_{f_j(\alpha_1 \ldots \widehat{\alpha_{k'}(i},ch,v,j))}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j))) \\
&= \widehat{\mathcal{D}}_{f_j(\widehat{i,ch},v,j)}(\widehat{\mathcal{D}}_{f_j(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j)))) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{\mathcal{D}}_{f_j(\widehat{\alpha_1 \ldots \alpha_{k'}})}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j)))) && \text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&\lesssim \widehat{\mathcal{D}}_{\widehat{ch?v}}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i))) && \text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\
&\not\lesssim \emptyset && \text{(by Lemma 5)}
\end{aligned}$$

Hence from Lemma 11 we get $\widehat{\mathcal{E}^j}, \widehat{\mathcal{X}^j} \vDash s''_j$ and $\rho''_j \in \gamma_{st}(\widehat{\mathcal{E}^j_\rho}(\mathit{first}(s''_j)))$.

Finally

$$\begin{aligned}
\widehat{\mathcal{D}}_{f_j(\alpha_1\ldots\widehat{\alpha_{k'}\cdot(i},ch,v,j))}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j))) &= \widehat{\mathcal{D}}_{f_j(\widehat{(i,ch},v,j)}(\widehat{\mathcal{D}}_{f_j(\widehat{\alpha_1\ldots\alpha_{k'}})}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j)))) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\text{(by def. of } \widehat{\mathcal{D}}\text{)} \\
&= \widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{D}}_{f_j(\widehat{\alpha_1\ldots\alpha_{k'}})}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s_j)))) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\text{(by def. of } f_j\text{)} \\
&\underset{\sim}{\sqsubset} \widehat{\mathcal{D}}_{\widehat{ch!v}}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s'_j))) \\
&\qquad\qquad\qquad\text{(by monotonicity of } \widehat{\mathcal{D}}\text{, IH)} \\
&= \widehat{\mathcal{D}}_{\widehat{\overline{ch?v}}}(\widehat{\mathcal{E}^j_f}(\mathit{first}(s'_j))) \quad \text{(by def. of } \bar{\cdot}\text{)} \\
&\underset{\sim}{\sqsubset} \widehat{\mathcal{E}^j_f}(\mathit{first}(s''_j)) \qquad\qquad \text{(by Lemma 11)}
\end{aligned}$$

□

B.4 Futures as histories, sans sum (Lemma 13)

Proof. By induction on the length of the trace. Let $s_1 : \cdots : s_n$, ρ_{init}, and an arbitrary trace $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$ be given.

case $k = 0$**:** First $\hbar_i(\epsilon) = \epsilon$ and $f_i(\epsilon) = \epsilon$ for any i. Let i be given. Then $\big|\big|_{j\neq i} \hbar_j(\epsilon) = \big|\big|_{j\neq i} \epsilon = \{\epsilon\}$

case $k = k' + 1$**:** Given a program and a trace $\langle s_1, \rho_{init}\rangle \ldots \langle s_n, \rho_{init}\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_{k'}}{\Longrightarrow} c'_1 \ldots c'_n \overset{\alpha_{k'+1}}{\Longrightarrow} c''_1 \ldots c''_n$ satisfying the above requirements, by the IH for any $c'_i = \langle s'_i, \rho'_i\rangle$ we have $f_i(\alpha_1 \ldots \alpha_{k'}) \in \big|\big|_{j\neq i} \hbar_j(\alpha_1 \ldots \alpha_{k'})$. We proceed by case analysis on the applied system rule.

case SYSTAU**:** By definition, for any i, j we have $\hbar_i(\alpha_1 \ldots \alpha_{k'} \cdot (j, \tau)) = \hbar_i(\alpha_1 \ldots \alpha_{k'})$ and $f_i(\alpha_1 \ldots \alpha_{k'} \cdot (j, \tau)) = f_i(\alpha_1 \ldots \alpha_{k'})$. Therefore

$$\begin{aligned}
f_i(\alpha_1 \ldots \alpha_{k'} \cdot (l, \tau)) &= f_i(\alpha_1 \ldots \alpha_{k'}) && \text{(by def. of } f_i\text{)} \\
&\in \Big|\Big|_{j\neq i} \hbar_j(\alpha_1 \ldots \alpha_{k'}) && \text{(by the IH)} \\
&= \Big|\Big|_{j\neq i} \hbar_j(\alpha_1 \ldots \alpha_{k'+1}) && \text{(by def. of } \hbar_j\text{)}
\end{aligned}$$

case SYSCOMM**:** $c'_1 \ldots c'_n \overset{i,ch,v,j}{\Longrightarrow} c''_1 \ldots c''_n$

There are now three sub-cases to consider:

Writer index i on LHS: We have $f_i(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) = f_i(\alpha_1 \dots \alpha_{k'}) \cdot ch?v$ and $\hbar_j(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) = \hbar_j(\alpha_1 \dots \alpha_{k'}) \cdot ch?v$. Therefore

$$
\begin{aligned}
& f_i(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \\
& = f_i(\alpha_1 \dots \alpha_{k'}) \cdot ch?v && \text{(by def. of } f_i\text{)} \\
& \in \left(\Big\Vert_{l \neq i} \hbar_l(\alpha_1 \dots \alpha_{k'}) \right) \cdot ch?v && \text{(by the IH, } \cdot \text{ monotone)} \\
& \subseteq \hbar_j(\alpha_1 \dots \alpha_{k'}) \cdot ch?v \parallel \Big\Vert_{l \neq i,j} \hbar_l(\alpha_1 \dots \alpha_{k'}) && \text{(by Lemma 3)} \\
& = \hbar_j(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \parallel \Big\Vert_{l \neq i,j} \hbar_l(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \\
& && \text{(by def. of } \hbar_l\text{)} \\
& = \Big\Vert_{l \neq i} \hbar_l(\alpha_1 \dots \alpha_{k'+1}) && \text{(simplify)}
\end{aligned}
$$

Reader index j on LHS: We have $f_j(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) = f_j(\alpha_1 \dots \alpha_{k'}) \cdot ch!v$ and $\hbar_i(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) = \hbar_i(\alpha_1 \dots \alpha_{k'}) \cdot ch!v$. Therefore

$$
\begin{aligned}
& f_j(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \\
& = f_j(\alpha_1 \dots \alpha_{k'}) \cdot ch!v && \text{(by def. of } f_j\text{)} \\
& \in \left(\Big\Vert_{l \neq j} \hbar_l(\alpha_1 \dots \alpha_{k'}) \right) \cdot ch!v && \text{(by the IH, } \cdot \text{ monotone)} \\
& \subseteq \hbar_i(\alpha_1 \dots \alpha_{k'}) \cdot ch!v \parallel \Big\Vert_{l \neq i,j} \hbar_l(\alpha_1 \dots \alpha_{k'}) && \text{(by Lemma 3)} \\
& = \hbar_i(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \parallel \Big\Vert_{l \neq i,j} \hbar_l(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \\
& && \text{(by def. of } \hbar_l\text{)} \\
& = \Big\Vert_{l \neq j} \hbar_l(\alpha_1 \dots \alpha_{k'+1}) && \text{(simplify)}
\end{aligned}
$$

Both indices i, j on RHS: When $l \notin \{i, j\}$ we have

$$\begin{aligned}
& f_l(\alpha_1 \dots \alpha_{k'} \cdot (i, ch, v, j)) \\
&= f_l(\alpha_1 \dots \alpha_{k'}) \cdot f_l(i, ch, v, j) && \text{(by def. of } f_l\text{)} \\
&\in \left(\bigg\Vert_{m \neq l} \hbar_m(\alpha_1 \dots \alpha_{k'}) \right) \cdot f_l(i, ch, v, j) && \text{(by the IH, } \cdot \text{ monotone)} \\
&\subseteq \left(\bigg\Vert_{m \neq l} \hbar_m(\alpha_1 \dots \alpha_{k'}) \right) \cdot (ch!v \parallel ch?v) \\
& && \text{(by def. of } f_l, \parallel, \cdot \text{ monotone)} \\
&\subseteq \hbar_i(\alpha_1 \dots \alpha_{k'}) \cdot ch!v \parallel \hbar_j(\alpha_1 \dots \alpha_{k'}) \cdot ch?v \parallel \bigg\Vert_{m \neq i,j,l} \hbar_m(\alpha_1 \dots \alpha_{k'}) \\
& && \text{(by Lemma 3)} \\
&= \hbar_i(\alpha_1 \dots \alpha_{k'}) \cdot ch!v \parallel \hbar_j(\alpha_1 \dots \alpha_{k'}) \cdot ch?v \parallel \bigg\Vert_{m \neq i,j,l} \hbar_m(\alpha_1 \dots \alpha_{k'+1}) \\
& && \text{(by def. of } \hbar_m\text{)} \\
&= \bigg\Vert_{m \neq l} \hbar_m(\alpha_1 \dots \alpha_{k'+1}) && \text{(simplify)}
\end{aligned}$$

□

B.5 Helper lemmas for history soundness (Lemma 15)

We first establish a couple of helper results:

Lemma 17 **(ϵ in prefix).** *For all $s, \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}$. $\epsilon \in \mathcal{L}(p)$ where $\langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$*

Proof. By structural induction on s. Let $s, \widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}$ be given. In the cases $\texttt{skip}^\ell$, $x =^\ell e$, $\texttt{for}\ b^\ell\ \{\, s \,\}$, $\texttt{select}^\ell \{\, a_1 \dots a_n \,\}$, $\texttt{case}\ x\ \texttt{=}\ \texttt{<-}^\ell\, ch\ \texttt{:}\ s$, and $\texttt{case}\ ch\ \texttt{<-}^\ell\, e\ \texttt{:}\ s$ the result holds immediately from the definition of $\mathcal{H}$. In the last two cases $s_1\ \texttt{;}\ s_2$ and $\texttt{if}\ b^\ell\ \{\, s_1 \,\}\ \texttt{else}\ \{\, s_2 \,\}$ it follows immediately from the induction hypothesis on s_1.

Lemma 18 **(Terminal action is τ).** *For all s, ρ, ρ', α. If $\langle s, \rho \rangle \xrightarrow{\alpha} \rho'$ then $\alpha = \tau$*

Proof. By case analysis on the applied rule for $\langle s, \rho \rangle \xrightarrow{\alpha} \rho'$. In the cases SKIP, ASSIGN, and FOR2 we immediately have $\alpha = \tau$ as desired. The cases SEQ1, SEQ2, IF1, IF2, FOR1, SELECT, READ, and WRITE can only transition to non-terminal configurations and are therefore vacuously true.

Lemma 19 **(ϵ in terminal action trace).** *For all $s, \rho, \rho', \alpha, \widehat{\mathcal{E}}, \widehat{\mathcal{X}}$. If $\langle s, \rho \rangle \xrightarrow{\alpha} \rho'$ and $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s$ then $\epsilon \in \mathcal{L}(c)$ where $\langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$.*

Proof. By the above lemma we have $\alpha = \tau$. We proceed by case analysis on the applied rule for $\langle s, \rho \rangle \xrightarrow{\alpha} \rho'$.

case Skip: By definition $\langle \epsilon, \epsilon \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{skip}^\ell)$ hence $\epsilon \in \mathcal{L}(\epsilon)$ as desired.
case Assign: By definition $\langle \epsilon, \epsilon \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, x =^\ell e)$ hence $\epsilon \in \mathcal{L}(\epsilon)$ as desired.
case For2: By definition $\langle c^* \cdot p, c^* \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{for}\ b^\ell\ \{\, s \,\})$ where $\langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$. hence $\epsilon \in \mathcal{L}(c^*)$ as desired.

The cases Seq1, Seq2, If1, If2, For1, Select, Read, and Write can only transition to non-terminal configurations and are therefore vacuously true.

Lemma 20 **($\mathcal{H}$ records actions).** *For all $s, s', \rho, \rho', \alpha, \widehat{\mathcal{E}}, \widehat{\mathcal{X}}$. If $\langle s, \rho \rangle \xrightarrow{\alpha} \langle s', \rho' \rangle$, $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s$, $\rho \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s)))$, $\rho' \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s')))$, and $\langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$ and $\langle p', c' \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s')$ then $|\alpha| \cdot \mathcal{L}(p') \subseteq \mathcal{L}(p)$ and $|\alpha| \cdot \mathcal{L}(c') \subseteq \mathcal{L}(c)$*

where we use an operation over single-process labels defined as follows:

$$|\tau| = \epsilon \quad |\alpha| = \alpha \ \text{ for } \ \alpha \neq \tau$$

Proof. By structural induction on s. We proceed by case analysis on the applied rule for $\langle s, \rho \rangle \xrightarrow{\alpha} \langle s', \rho' \rangle$.

case Seq1: We have $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1\ \texttt{;}\ s_2) = \langle p_1 + c_1 \cdot p_2, c_1 \cdot c_2 \rangle$ where $\langle p_1, c_1 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1)$ and $\langle p_2, c_2 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_2)$.
Furthermore since $\langle s_1, \rho \rangle \xrightarrow{\alpha} \langle s_3, \rho' \rangle$. we have $|\alpha| \cdot \mathcal{L}(p_3) \subseteq \mathcal{L}(p_1)$ and $|\alpha| \cdot \mathcal{L}(c_3) \subseteq \mathcal{L}(c_1)$ by the induction hypothesis. Hence $|\alpha| \cdot \mathcal{L}(p_3 + (c_3 \cdot p_2) + (c_3 \cdot c_2)) \subseteq \mathcal{L}(p_1 + (c_1 \cdot p_2) + (c_1 \cdot c_2))$ as desired.
case Seq2: By the above lemma we have $\alpha = \tau$. Furthermore $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1\ \texttt{;}\ s_2) = \langle p_1 + c_1 \cdot p_2, c_1 \cdot c_2 \rangle$ where $\langle p_1, c_1 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1)$ and $\langle p_2, c_2 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_2)$. and by the above lemma we have $\epsilon \in \mathcal{L}(c_1)$. Therefore since $|\tau| = \epsilon$ we have $|\tau| \cdot \mathcal{L}(p_2) = \mathcal{L}(p_2) \subseteq \mathcal{L}(p_1 + c_1 \cdot p_2)$ and $|\tau| \cdot \mathcal{L}(c_2) = \mathcal{L}(c_2) \subseteq \mathcal{L}(c_1 \cdot c_2)$ as desired.
case If1: We have $\alpha = \tau$. Since $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{if}\ ^\ell\ \{\, s \,\}\ \texttt{else}\ \{\ \}) = \langle p_1 + p_2, c_1 + c_2 \rangle$ where $\langle p_1, c_1 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_1)$ and $\langle p_2, c_2 \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_2)$ we therefore have $|\tau| = \epsilon$ and hence $|\tau| \cdot \mathcal{L}(p_1) = \mathcal{L}(p_1) \subseteq \mathcal{L}(p_1 + p_2)$ and $|\tau| \cdot \mathcal{L}(c_1) = \mathcal{L}(c_1) \subseteq \mathcal{L}(c_1 + c_2)$ as desired.
case If2: Symmetric to If2.
case For1: We have $\alpha = \tau$. Since $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{for}\ b^\ell\ \{\, s \,\}) = \langle c^* \cdot p, c^* \rangle$ where $\langle p, c \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$ and $s' = s\ \texttt{;}\ \texttt{for}\ b^\ell\ \{\, s \,\}$ we thus have $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s\ \texttt{;}\ \texttt{for}\ b^\ell\ \{\, s \,\}) = \langle p + (c \cdot c^* \cdot p), c \cdot c^* \rangle$. Hence $|\tau| = \epsilon$ and therefore $|\tau| \cdot \mathcal{L}(p + (c \cdot c^* \cdot p)) = \mathcal{L}(p + (c \cdot c^* \cdot p)) \subseteq \mathcal{L}(c^* \cdot p)$ and $|\tau| \cdot \mathcal{L}(c \cdot c^*) = \mathcal{L}(c \cdot c^*) \subseteq \mathcal{L}(c^*)$ as desired.
case Select: We have $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{select}^\ell\ \{\, a_1 \ldots a_n \,\}) = \langle \epsilon + \sum_j p_j, \sum_j c_j \rangle$ where $\langle p_j, c_j \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, a_j)$ for $1 \leq j \leq n$. Since $\langle a_j, \rho \rangle \xrightarrow{\alpha} \langle s', \rho' \rangle$. by the induction hypothesis we have $|\alpha| \cdot \mathcal{L}(p') \subseteq \mathcal{L}(p_j)$ and $|\alpha| \cdot \mathcal{L}(c') \subseteq \mathcal{L}(c_j)$ and where $\langle p', c' \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s')$. But then $|\alpha| \cdot \mathcal{L}(p') \subseteq \mathcal{L}(p_j) \subseteq \mathcal{L}(\epsilon + \sum_j p_j)$ and $|\alpha| \cdot \mathcal{L}(c') \subseteq \mathcal{L}(c_j) \subseteq \mathcal{L}(\sum_j c_j)$ as desired.

case READ: We have $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{case}\ x = \texttt{<-}^\ell\ ch : s) = \langle \epsilon + ch?\widehat{v} + ch?\widehat{v} \cdot p, ch?\widehat{v} \cdot c\rangle$.
where $\langle p, c\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$ and $\widehat{v} = \widehat{\mathcal{E}}^i_\rho(\mathit{first}(s))(x)$.
By assumption $\rho' = \rho[x \mapsto v] \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s)))$. Hence $v \in \gamma_v(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s))(x))$ and therefore $ch?v \in \mathcal{L}(ch?\widehat{v})$. But then we have $|ch?v| \cdot \mathcal{L}(p) = ch?v \cdot \mathcal{L}(p) \subseteq \mathcal{L}(\epsilon + ch?\widehat{v} + ch?\widehat{v} \cdot p)$ and $|ch?v| \cdot \mathcal{L}(c) = ch?v \cdot \mathcal{L}(c) \subseteq \mathcal{L}(ch?\widehat{v} \cdot c)$ as desired.

case WRITE: We have $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, \texttt{case}\ ch\ \texttt{<-}^\ell\ e : s) = \langle \epsilon + ch!\widehat{v} + ch!\widehat{v} \cdot p, ch!\widehat{v} \cdot c\rangle$.
where $\langle p, c\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s)$ and $\widehat{v} = \widehat{\mathcal{A}}(e, \widehat{\mathcal{E}}^i_\rho(\ell))$.
By Lemma 9 we have $v \in \gamma_v(\widehat{v})$ and therefore $ch!v \in \mathcal{L}(ch!\widehat{v})$. But then $|ch!v| \cdot \mathcal{L}(p) = ch!v \cdot \mathcal{L}(p) \subseteq \mathcal{L}(\epsilon + ch!\widehat{v} + ch!\widehat{v} \cdot p)$ and $|ch!v| \cdot \mathcal{L}(c) = ch!v \cdot \mathcal{L}(c) \subseteq \mathcal{L}(ch!\widehat{v} \cdot c)$ as desired.

The cases SKIP, ASSIGN, and FOR2 are vacuously true as they transition to a terminal configuration.

B.6 History soundness (Lemma 15)

We now address the main history lemma.

Proof. By induction on the length of the trace. Let a program $s_1 : \cdots : s_n$, stores $\rho_1, \ldots, \rho_n$, a trace $\langle s_1, \rho_1\rangle \ldots \langle s_n, \rho_n\rangle \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} c'_1 \ldots c'_n$, $1 \leq i \leq n$, and caches $\widehat{\mathcal{E}}, \widehat{\mathcal{X}}$ be given such that $\rho_i \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s_i)))$, $f_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s_i)))$, and $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_i$.

case $k = 0$: We have $\hbar_i(\epsilon) = \epsilon$. Furthermore we have $\epsilon \in \mathcal{L}(p_i) \subseteq \mathcal{L}(p_i + c_i)$ for $\langle p_i, c_i\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i)$ by Lemma 17 above.

case $k = k' + 1$: We have $\langle s_1, \rho_1\rangle \ldots \langle s_n, \rho_n\rangle \overset{\alpha_1}{\Longrightarrow} \langle s'_1, \rho'_1\rangle \ldots \langle s'_n, \rho'_n\rangle$ and by (the generalization of) Theorem 12 we furthermore have $\rho'_i \in \gamma_{st}(\widehat{\mathcal{E}}^i_\rho(\mathit{first}(s'_i)))$, $f_i(\alpha_2 \ldots \alpha_k) \in \mathcal{L}(\widehat{\mathcal{E}^i_f}(\mathit{first}(s'_i)))$, and $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s'_i$. We proceed by case analysis on α_1.

case $\alpha_1 = \langle j, \tau\rangle, j \neq i$: We have $s_i = s'_i$. Therefore $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i) = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$. Furthermore $\hbar_i(\alpha_1 \ldots \alpha_k) = \hbar_i(\alpha_2 \ldots \alpha_k) \in \mathcal{L}(p_i + c_i)$ with $\langle p_i, c_i\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i) = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$ by the induction hypothesis.

case $\alpha_1 = \langle i, \tau\rangle$: If $\langle s_i, \rho_i\rangle \xrightarrow{\tau} \rho'$: We immediately have $\epsilon \in \mathcal{L}(p)$ where $\langle p, c\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i)$ by Lemma 17 above.
Now assume $\hbar_i(\alpha_2 \ldots \alpha_k) \neq \epsilon$. Then some $\hbar_i(\alpha_w) \neq \epsilon$ which means $\alpha_w = (l, ch, v, j)$ with either $l = i$ or $j = i$. But then there should exist a process transition from ρ' which is impossible. Hence we have $\hbar_i(\alpha_1 \ldots \alpha_k) = \hbar_i(\alpha_2 \ldots \alpha_k) = \epsilon \in \mathcal{L}(p_i + c_i)$ as desired.
If $\langle s_i, \rho_i\rangle \xrightarrow{\tau} \langle s'_i, \rho'\rangle$ we have $\hbar_i(\langle i, \tau\rangle \alpha_2 \ldots \alpha_k) = \hbar_i(\alpha_2 \ldots \alpha_k) \in \mathcal{L}(p'_i + c'_i)$ with $\langle p'_i, c'_i\rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$ by the induction hypothesis. Hence by Lemma 20 above $|\tau| \cdot \mathcal{L}(p'_i + c'_i) = \mathcal{L}(p'_i + c'_i) \subseteq \mathcal{L}(p_i + c_i)$ and we conclude $\hbar_i(\alpha_1 \ldots \alpha_k) \in \mathcal{L}(p_i + c_i)$ as desired.

case $\alpha_1 = l, ch, v, j$ **for** $i \notin \{l, j\}$**:** We have $s_i = s'_i$. Therefore $\mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i) = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$. Furthermore $\hbar_i((l, ch, v, j)\alpha_2 \dots \alpha_k) = \hbar_i(\alpha_2 \dots \alpha_k) \in \mathcal{L}(p_i + c_i)$ with $\langle p_i, c_i \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s_i) = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$ by the induction hypothesis.

case $\alpha_1 = i, ch, v, j$**:** By rule SysComm it must be the case that $\langle s_i, \rho \rangle \xrightarrow{ch!v} \langle s'_i, \rho' \rangle$ since if $\langle s_i, \rho \rangle \xrightarrow{ch!v} \rho'$ we would conclude $ch!v = \tau$ by Lemma 19 above and reach a contradiction.
But then $\hbar_i(i, ch, v, j) = ch!v$ and by Lemma 20 above we have $|ch!v| \cdot \mathcal{L}(p'_i) = ch!v \cdot \mathcal{L}(p'_i) \subseteq \mathcal{L}(p_i)$ and $|ch!v| \cdot \mathcal{L}(c'_i) = ch!v \cdot \mathcal{L}(c'_i) \subseteq \mathcal{L}(c_i)$ where $\langle p'_i, c'_i \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$. Furthermore, by the induction hypothesis $\hbar_i(\alpha_2 \dots \alpha_k) \in \mathcal{L}(p'_i + c'_i)$ hence $\hbar_i(\alpha_1\alpha_2 \dots \alpha_k) = ch!v\hbar_i(\alpha_2 \dots \alpha_k) \in ch!v \cdot \mathcal{L}(p'_i + c'_i) \subseteq \mathcal{L}(p_i + c_i)$ as desired.

case $\alpha_1 = l, ch, v, i$**:** By rule SysComm it must be the case that $\langle s_i, \rho \rangle \xrightarrow{ch?v} \langle s'_i, \rho' \rangle$ since if $\langle s_i, \rho \rangle \xrightarrow{ch?v} \rho'$ we would conclude $ch?v = \tau$ by Lemma 19 above and reach a contradiction.
But then $\hbar_i(l, ch, v, i) = ch?v$ and by Lemma 20 above we have $|ch?v| \cdot \mathcal{L}(p'_i) = ch?v \cdot \mathcal{L}(p'_i) \subseteq \mathcal{L}(p_i)$ and $|ch?v| \cdot \mathcal{L}(c'_i) = ch?v \cdot \mathcal{L}(c'_i) \subseteq \mathcal{L}(c_i)$ where $\langle p'_i, c'_i \rangle = \mathcal{H}(\widehat{\mathcal{E}^i}, \widehat{\mathcal{X}^i}, s'_i)$. Furthermore, by the induction hypothesis $\hbar_i(\alpha_2 \dots \alpha_k) \in \mathcal{L}(p'_i + c'_i)$ hence $\hbar_i(\alpha_1\alpha_2 \dots \alpha_k) = ch?v\hbar_i(\alpha_2 \dots \alpha_k) \in ch?v \cdot \mathcal{L}(p'_i + c'_i) \subseteq \mathcal{L}(p_i + c_i)$ as desired.