This is the authors' full version including proofs. The original version appeared in the 18th International Symposium on Principles and Practice of Declarative Programming, 2016 (PPDP'16)

# **Iterated Process Analysis over** Lattice-Valued Regular Expressions

Jan Midtgaard

DTU Compute Technical University of Denmark mail@janmidtgaard.dk

Flemming Nielson

DTU Compute Technical University of Denmark fnie@dtu.dk

Hanne Riis Nielson

DTU Compute Technical University of Denmark hrni@dtu.dk

# Abstract

We present an iterated approach to statically analyze programs of two processes communicating by message passing. Our analysis operates over a domain of lattice-valued regular expressions, and computes increasingly better approximations of each process's communication behavior. Overall the work extends traditional semantics-based program analysis techniques to automatically reason about message passing in a manner that can simultaneously analyze both values of variables as well as message order, message content, and their interdependencies.

Categories and Subject Descriptors F.3.2 [Logics and Meanings of Programs]: Semantics of Programming Languages-Program Analysis

General Terms Languages, Theory, Verification

Keywords Process analysis, abstract interpretation, lattice-valued regular expressions

# 1. Introduction

Today's software increasingly depend on network communication, as witnessed, e.g., by the popularity of applications for smartphones and the web. This network communication can depend on intricate protocol details, such as the order and the content of messages. To this end we develop an 'interaction analysis': a static analysis that infers (an approximation of) both the structure of interaction as well as the content.

```
spawn proc1() { x = 1;
                while (0 < x) \{ ch?x \} \}
spawn proc2() { y = 1000;
                 while (0 < y) \{ ch!y;
                                  y = y-1; \} \}
```

Figure 1: An example program

Consider the program in Fig. 1. It consists of two processes communicating by message passing. The first process proc1, repeatedly reads an integer from a channel ch. A second process proc2, writes a decreasing chain of integers to channel ch.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

PPDP '16, September 05-07, 2016, Edinburgh, United Kingdom

Our analysis captures the interaction between the two processes by means of lattice-valued regular expressions (Midtgaard et al. 2016). In doing so, it automatically infers

- $(ch![1;1000])^*$  as a communication invariant for proc2 (here expressed as a regular expression over intervals annotated with channel names), meaning that proc2 repeatedly outputs an integer value in the range [1; 1000] to the channel ch,
- that the loop of proc2 terminates with the value of y being approximated by the interval [0; 0], and
- that the loop of proc1 does not terminate, since none of the values received suffice to falsify its loop condition.

The contributions of this paper are

- a static interaction analysis for inferring both the communication structure and content of two message passing processes,
- an iterative algorithm for computing it,
- a prototype implementation of the approach, and
- a soundness proof of the developed analysis.

# 2. Lattice Theory and Abstract Interpretation

In this section we summarize the required background material on lattice theory (Grätzer 1978; Davey and Priestley 2002) and abstract interpretation (Cousot and Cousot 1977, 1992a) which the rest of the paper builds on.

A lattice  $\langle L; \sqsubseteq \rangle$  is a partially ordered set where each pair of elements  $\ell, \ell' \in L$  has a least upper bound  $\ell \sqcup \ell'$  and a greatest lower bound  $\ell \sqcap \ell'$ . A complete lattice  $\langle L; \sqsubseteq \rangle$  is a partially ordered set where a least upper bound  $\sqcup S$  and greatest lower bound  $\sqcap S$ exists for any subset of elements  $S \subseteq L$ . In particular this means that L has a least element  $\bot = \Box L = \sqcup \emptyset$  ('bottom') and a greatest element  $\top = \Box L = \Box \emptyset$  ('top'). A Moore family of a partially ordered set  $\langle L; \sqsubseteq \rangle$  with  $\top \in L$ , is a subset  $S \subseteq L$  such that  $\top \in S$ and  $\sqcap X \in S$  for any subset  $X \subseteq S$ .

A Galois connection  $\langle C; \sqsubseteq \rangle \xrightarrow{\sim} \langle A; \leq \rangle$  is a pair of functions  $\alpha : C \longrightarrow A$  and  $\gamma : A \longrightarrow C$  connecting two partially ordered sets such that  $\forall a \in A, c \in C. \ \alpha(c) \leq a \iff c \sqsubseteq \gamma(a).$ The latter is equivalent to requiring that  $\alpha$  and  $\gamma$  are monotone  $(\forall c, c' \in C. \ c \sqsubseteq c' \implies \alpha(c) \le \alpha(c') \text{ and } \forall a, a' \in A. \ a \le a' \in C.$  $a' \implies \gamma(a) \sqsubseteq \gamma(a')$ , that  $\alpha \circ \gamma$  is reductive  $(\forall a. (\alpha \circ \gamma)(a) \le a)$ , and that  $\gamma \circ \alpha$  is extensive  $(\forall c. \ c \sqsubseteq (\gamma \circ \alpha)(c))$ . A Galois insertion  $\langle C; \Box \rangle \xleftarrow{\gamma}{\alpha} \langle A; \leq \rangle$  is a Galois connection where  $\alpha$  is surjective, or equivalently (1) that  $\gamma$  is injective or (2) that  $\alpha \circ \gamma$  is the identity function. When a Galois connection  $\langle C; \sqsubseteq \rangle \xrightarrow{\gamma} \langle A; \leq \rangle$  connects two complete lattices  $\alpha$  is a complete join morphism, meaning that

Copyright C: 2016 ACM 978-1-4503-4148-6/16/09... \$15.00 DOI: http://dx.doi.org/10.1145/2967973.2968601 Reprinted from PPDP '16,, Proceedings, September 05-07, 2016, Edinburgh, United Kingdom, pp. 1–27.

 $\alpha(\bigsqcup_i \ell_i) = \bigvee_i \alpha(\ell_i)$  and  $\gamma$  is a complete meet morphism, meaning that  $\gamma(\bigwedge_i \ell_i) = \prod_i \gamma(\ell_i)$ .

An atom of a lattice  $\langle L; \sqsubseteq \rangle$  is an element  $a \in L$  such that if another element  $\ell \in L$  satisfies  $\bot \sqsubseteq \ell \sqsubseteq a$  then  $\ell = \bot$  or  $\ell = a$ . We write Atoms(L) for the set of L's atom elements and let variables a, a' range over these. An atomic lattice is a lattice  $\langle L; \sqsubseteq \rangle$  where for any non-bottom element  $\ell \in L$  there exists an atom  $a \in Atoms(L)$  such that  $a \sqsubseteq \ell$ . An atomistic lattice is a lattice where for any non-bottom element  $\ell \in L$  there exists a set of atoms  $S_{\ell} \subseteq Atoms(L)$  such that  $\ell = \sqcup S_{\ell}$ . An atomistic Galois insertion is a Galois insertion connecting two atomistic lattices, such that  $\alpha : Atoms(C) \longrightarrow Atoms(A)$  is surjective, i.e.,  $\alpha$ maps atoms to atoms and for all  $a \in Atoms(A)$  there exists an atom  $c \in Atoms(C)$  such that  $\alpha(c) = a$ .

A fixed point of a function  $F: L \longrightarrow L$  is an element  $\ell \in L$ such that  $F(\ell) = \ell$ . A post-fixed point of F is an element  $\ell \in L$ such that  $F(\ell) \sqsubseteq \ell$ .<sup>1</sup> Tarski's fixed point theorem says that for a complete lattice  $\langle L; \sqsubseteq \rangle$  and a monotone function  $F: L \longrightarrow L$  the fixed points of F themselves form a complete lattice, i.e., the set of fixed points is non-empty, and in particular there exists a least fixed point lfp F and a greatest fixed point gfp F.

In classical abstract interpretation ("the Galois connection framework") the fixed point transfer theorem ties together a Galois connection  $\langle C; \sqsubseteq \rangle \xrightarrow{\langle \gamma \rangle} \langle A; \leq \rangle$ , two monotone functions  $F: C \longrightarrow C, \hat{F}: A \longrightarrow A$ , and least fixed points lfp F, lfp $(\alpha \circ F \circ \gamma)$ , and lfp  $\hat{F}$  by concluding that if  $\hat{F}$  over-approximates  $\alpha \circ F \circ \gamma \ (\forall a \in A. \ (\alpha \circ F \circ \gamma)(a) \leq \hat{F}(a))$  then we can over-approximate a fixed point of F by any monotone overapproximation of  $\alpha \circ F \circ \gamma: \alpha(\text{lfp } F) \leq \text{lfp}(\alpha \circ F \circ \gamma) \leq \text{lfp } \hat{F}$ .

If  $\langle A; \leq \rangle$  satisfies the ascending chain condition (ACC) meaning that all strictly increasing chains have finite length, then we can compute  $\operatorname{lfp} \widehat{F}$  by Kleene iteration as the limit of  $\bot = \widehat{F}^0(\bot), \widehat{F}^1(\bot), \widehat{F}^2(\bot), \ldots$  If  $\langle A; \leq \rangle$  does not satisfy the ACC (or if it takes too long to compute the above), then we can overapproximate  $\operatorname{lfp} \widehat{F}$  further by computing an iteration with widening. The alternative iteration sequence is computed as  $\ell_0 = \bot, \ \ell_1 = \ell_0 \nabla \widehat{F}(\ell_0), \ \ell_2 = \ell_1 \nabla \widehat{F}(\ell_1), \ \ldots$  where the widening operator  $\nabla : A \longrightarrow A \longrightarrow A$  is defined as follows:

Definition 2.1 (Widening). A widening operator satisfies

1.  $\forall \ell, \ell' \in A. \ \ell \sqsubseteq \ell \bigtriangledown \ell' \land \ell' \sqsubseteq \ell \lor \ell'$ 

2. for all increasing chains  $\ell_0 \sqsubseteq \ell_1 \sqsubseteq \ell_2 \sqsubseteq \ldots$  the alternative chain defined as  $x_0 = \ell_0$  and  $x_{k+1} = x_k \lor \ell_{k+1}$  stabilizes after a finite number of steps.

There are multiple definitions of widening in the literature. The definition given above is a common one (Cousot and Cousot 1992b; Halbwachs 1993; Nielson et al. 1999).  $^2$ 

To improve upon a post-fixed point  $\ell_{post}$ , e.g., found by the above iteration, one can compute a second iteration with narrowing  $\ell'_0 = \ell_{post}, \ \ell'_1 = \ell'_0 \triangle \widehat{F}(\ell'_0), \ \ell'_2 = \ell'_1 \triangle \widehat{F}(\ell'_1), \ \dots$  where the narrowing operator  $\triangle : A \longrightarrow A \longrightarrow A$  is defined as follows:

Definition 2.2 (Narrowing). A narrowing operator satisfies

$$\begin{split} E \ni e &::= n \mid x \mid ? \mid e_1 + e_2 \mid e_1 - e_2 \\ B \ni b &::= \text{tt} \mid \text{ff} \mid x_1 < x_2 \\ S \ni s &::= \text{skip}^{\ell} \mid x :=^{\ell} e \mid s_1; s_2 \mid \text{if} \ b^{\ell} \text{ then } s_1 \text{ else } s_2 \\ \mid \text{while} \ b^{\ell} \text{ do } s_1 \text{ end} \mid s_1 \oplus^{\ell} s_2 \mid ch?^{\ell} x \mid ch!^{\ell} e \mid \text{stop}^{\ell} \\ P \ni p &::= s_1 \parallel s_2 \end{split}$$

Figure 2: BNF syntax of the process language

- 1.  $\forall \ell, \ell' \in A. \ \ell' \sqsubseteq \ell \implies \ell' \sqsubseteq (\ell \land \ell') \sqsubseteq \ell$
- 2. for all decreasing chains  $\ell_0 \supseteq \ell_1 \supseteq \ell_2 \supseteq \ldots$  the alternative chain defined as  $x_0 = \ell_0$  and  $x_{k+1} = x_k \bigtriangleup \ell_{k+1}$  stabilizes after a finite number of steps.

A post-fixed point of an operator  $\widehat{F} : L \longrightarrow L$  typically represents a conservative static analysis answer. As such, rather than specifying an analysis as an explicit functional  $\widehat{F}$  over a lattice structure, one may instead give a specification of a post-fixed point (an 'acceptability relation'  $\widehat{F}(\ell) \sqsubseteq \ell$ ), thereby decoupling the specification of valid analysis answers from the means used to compute them. Both the syntax-directed *flow logic* (Nielson and Nielson 2002) and *constraint-based analyses* (Aiken 1999) share this approach.

# 3. Language

Our starting point is a core imperative language with sequencing, conditionals, and while loops as outlined in Fig. 2. The core language is structured into three syntactic categories of arithmetic expressions (e), Boolean expressions (b), and statements (s). For presentational purposes we keep the arithmetic and Boolean expressions minimal. The statements of the core language have been extended with primitives for non-deterministic choice  $(\oplus)$ , for reading and writing messages from/to a named channel (ch?x) and ch!e), and for terminating a process (stop). The two message passing primitives are synchronous. To build systems of communicating processes, we extend the language further with a syntactic category of programs (p), consisting of a pair of processes. To later state and prove soundness of our analysis all basic actions (basic statements or branch points in a corresponding flow graph) of the language have been labeled with labels  $\ell$  drawn from a set *Labels* to distinguish multiple occurrences of syntactically identical entities, e.g.,  $skip^1$  and  $skip^2$ . We assume that initially no two actions are labeled with the same label (during statement rewriting in the following SOS, we may duplicate sub-statements and hence encounter some label overlapping).

We provide an operational semantics for the language in Fig. 3. Following the syntactic structure of the language, it is formulated as two big-step evaluation relations for (non-deterministically) evaluating arithmetic expressions and Boolean expressions in a given store, and two small-step evaluation relations  $\longrightarrow$  and  $\Longrightarrow$  for executing statements and programs, respectively. The big-step evaluation relations of the arithmetic expressions and Boolean expressions are standard, with the exception of the arithmetic expression '?'. By the rule ANY in Fig. 3. such an arithmetic expression can evaluate to any value v. As traditional, communication between processes is modeled by labels on the transitions: the transition  $\xrightarrow{ch?v}$  signals a message read of a value  $v, \xrightarrow{ch!v}$  signals a message write of a value v, and  $\tau$  signals absence of a communication event (a process-local computation step). The label  $\alpha$  stands for any (potentially absent) communication event. A choice statement can non-deterministically execute either its left-hand or righthand statement depending on the available communication events. A stop-statement on the other hand halts the enclosing process: by

<sup>&</sup>lt;sup>1</sup>Note: Here the abstract interpretation literature (Cousot 1981; Cousot and Cousot 1992a) deviates from other literature (Davey and Priestley 2002; Sangiorgi 2009) where the above would be called a 'pre-fixed point'.

<sup>&</sup>lt;sup>2</sup> Cousot and Cousot (1976, 1977) initially state the first requirement as  $\forall \ell, \ell' \in A$ .  $\ell \sqcup \ell' \sqsubseteq \ell \bigtriangledown \ell' \lor \ell'$  which is equivalent to the above over a lattice structure with (binary) least upper bounds. Both Cousot and Cousot (1976, 1977) and the definition recalled by Monniaux (2009) strengthen the last requirement to hold for *any* sequence  $\ell_0, \ell_1, \ell_2, \ldots$  (regardless of order). Both Cousot and Cousot (1992a) and Cousot (2015) further adjust the above requirements.

$$\begin{array}{c} \overline{\rho \vdash_{\mathcal{A}} n \downarrow n} \ \text{Lit} & \overline{\rho \vdash_{\mathcal{A}} x \downarrow \rho(x)} \ \text{Var} & \overline{\rho \vdash_{\mathcal{A}}? \downarrow v} \ \text{Any} \\ \hline \frac{\rho \vdash_{\mathcal{A}} e_1 \downarrow v_1 \quad \rho \vdash_{\mathcal{A}} e_2 \downarrow v_2}{\rho \vdash_{\mathcal{A}} e_1 + e_2 \downarrow v_1 + v_2} \ \text{Add} \\ \hline \frac{\rho \vdash_{\mathcal{A}} e_1 \downarrow v_1 \quad \rho \vdash_{\mathcal{A}} e_2 \downarrow v_2}{\rho \vdash_{\mathcal{A}} e_1 - e_2 \downarrow v_1 - v_2} \ \text{Sub} & \overline{\rho \vdash_{\mathcal{B}} \text{tt} \downarrow \text{tt}} \ \text{True} \\ \hline \frac{\rho \vdash_{\mathcal{A}} e_1 \downarrow v_1 \quad \rho \vdash_{\mathcal{A}} e_2 \downarrow v_2}{\rho \vdash_{\mathcal{B}} x_1 < x_2 \downarrow \text{tt}} \ \text{LessThan1} \\ \hline \frac{\rho(x_1) \geq \rho(x_2)}{\rho \vdash_{\mathcal{B}} x_1 < x_2 \downarrow \text{tf}} \ \text{LessThan2} \\ \hline \frac{\rho(x_1) \geq \rho(x_2)}{\rho \vdash_{\mathcal{B}} x_1 < x_2 \downarrow \text{tf}} \ \text{LessThan2} \\ \hline \frac{\phi(x_1) \geq \rho(x_2)}{(s_1; s_2, \rho) \xrightarrow{\alpha} (s_3; p')} \ \text{SEQ1} \quad \frac{(s_1, \rho) \xrightarrow{\alpha} \rho'}{(s_1; s_2, \rho) \xrightarrow{\alpha} (s_2, \rho')} \ \text{SeQ2} \\ \hline \frac{(s_1, \rho) \xrightarrow{\alpha} (s_3, \rho')}{(s_1; s_2, \rho) \xrightarrow{\alpha} (s_3; s_2, \rho')} \ \text{SEQ1} \quad \frac{(s_1, \rho) \xrightarrow{\alpha} (s_2, \rho')}{(s_1; s_2, \rho) \xrightarrow{\alpha} (s_2, \rho')} \ \text{IF1} \\ \hline \frac{\rho \vdash_{\mathcal{B}} b \downarrow \text{tt}}{(\text{if } b^\ell \text{ then } s_1 \text{ else } s_2, \rho) \xrightarrow{\tau} (s_1, \rho)} \ \text{IF1} \\ \hline \frac{\rho \vdash_{\mathcal{B}} b \downarrow \text{tt}}{(shile b^\ell \text{ do } s_1 \text{ end}, \rho) \xrightarrow{\tau} (s_1; \text{while } b^\ell \text{ do } s_1 \text{ end}, \rho)} \ \text{WHILE1} \\ \hline \frac{\rho \vdash_{\mathcal{B}} b \downarrow \text{tf}}{(shile b^\ell \text{ do } s_1 \text{ end}, \rho) \xrightarrow{\tau} (s_1; \text{while } b^\ell \text{ do } s_1 \text{ end}, \rho)} \ \text{WHILE2} \\ \hline \frac{(ch?^\ell x, \rho) \xrightarrow{ch? p} \rho[x \mapsto v]}{(s_1, \phi^1) = \rho[x]} \ \text{READ} \quad \frac{\rho \vdash_{\mathcal{A}} e \downarrow v}{(ch!^\ell e_1, \rho) \xrightarrow{ch? p} \varphi} \ \text{WRITE} \\ \hline \frac{(s_1, \rho_1) \xrightarrow{\tau} c_1}{(s_1, \phi^1) \xrightarrow{\tau} c_1} \ \text{CHOICE1} \quad \frac{(s_2, \rho) \xrightarrow{\alpha} c_2}{(s_1 \oplus \ell^1 s_2, \rho) \xrightarrow{\alpha} c_2} \ \text{CHOICE2} \\ \hline \frac{(s_1, \rho_1) \xrightarrow{ch? p} c_1'}{(s_1, \phi^1) = (s_2, \rho_2) \xrightarrow{ch? p} c_1'} \ \text{SYSLEFT} \\ \hline \frac{(s_1, \rho_1) \xrightarrow{ch? p} c_1'}{(s_1, \rho_1) \parallel (s_2, \rho_2) \xrightarrow{ch? p} c_1'} \ \frac{(s_1, \rho_1) \xrightarrow{ch? p} c_1'}{(s_1, \rho_1) \parallel (s_2, \rho_2) \xrightarrow{ch? p} c_1'} \ \frac{s_2}{s_1} \ \text{SYSWR} \\ \hline \frac{(s_1, \rho_1) \xrightarrow{ch? p} c_1'}{(s_1, \rho_2) \xrightarrow{ch? p} c_1'} \ \frac{s_2}{s_1} \ \text{SYSRW} \\ \hline \end{array}$$

Figure 3: Operational semantics of the process language

omitting an explicit stop-rule from the semantics we model that such statements cannot continue execution.

In the four rules for system (or program) steps we annotate the arrow  $\implies$  with a pair of events from each process. In SYSWR for example,  $\stackrel{ch!v,ch?v}{\implies}$  expresses that the first process takes a step writing a value v to channel ch and that the second process reads v

$$\mathcal{L}(\emptyset) = \emptyset \qquad \qquad \mathcal{L}(r^*) = \bigcup_{i \ge 0} \mathcal{L}(r)^i$$
$$\mathcal{L}(\epsilon) = \{\epsilon\} \qquad \qquad \mathcal{L}(\mathbb{C}r) = \wp(C^*) \setminus \mathcal{L}(r)$$
$$\mathcal{L}(\ell) = \{c \mid c \in \gamma(\ell)\} \qquad \qquad \mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$$
$$\mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2) \qquad \qquad \mathcal{L}(r_1 \& r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2)$$
Figure 4: Denotation of LREs:  $\mathcal{L} : \widehat{R}_A \longrightarrow \wp(C^*)$ 

from *ch*. In the rules SYSLEFT and SYSRIGHT where only one of the two processes takes a step, we use an  $\epsilon$  to indicate that the other process takes no computation step. For example, in SYSLEFT  $\frac{\tau, \epsilon}{\tau}$  indicates that the first process takes a  $\tau$  step (no communication) while the second process performs no computation. Finally the system has no shared state: A process is limited to its own private store  $\rho$ .

Based on the operational semantics we can now characterize a system-level computation as a trace:

$$c_1, c_1' \stackrel{\alpha_1, \alpha_1'}{\Longrightarrow} \langle c_2, c_2' \rangle \stackrel{\alpha_2, \alpha_2'}{\Longrightarrow} \langle c_3, c_3' \rangle \stackrel{\alpha_3, \alpha_3'}{\Longrightarrow} \dots \stackrel{\alpha_{n-1}, \alpha_{n-1}'}{\Longrightarrow} \langle c_n, c_n' \rangle$$

where each configuration  $c_i$  (and  $c'_i$ ) is either a pair  $\langle s, \rho \rangle$  or a final configuration  $\rho$ . Intuitively for, e.g.,  $c_3 = \langle s_3, \rho_3 \rangle$  the string  $\alpha_1 \alpha_2$  represents the history of the first process's communication after two computation steps, whereas the string  $\alpha'_3 \dots \alpha'_{n-1}$  represents the future communication of the environment (the second process). As illustrated by our introductory example the analysis will approximate such strings by lattice-valued regular expressions.

# 4. Analysis

In this section we recall the domain of lattice-valued regular expressions and then develop the analysis based on this domain.

# 4.1 Lattice-valued regular expressions

Lattice-valued regular expressions (LREs) is a parametric abstract domain capable of expressing both values (ranging over a given lattice) as well as order, choice, and iteration of events (Midtgaard et al. 2016). Given a Galois insertion  $\langle \wp(C); \subseteq \rangle \xleftarrow{\gamma}{\alpha} \langle A; \equiv \rangle$  connecting an abstract domain A to some set of concrete characters where we furthermore require that  $\alpha : Atoms(\wp(C)) \longrightarrow Atoms(A)$ , we can define the A-valued LREs as follows.

$$\widehat{R}_A ::= \emptyset \mid \epsilon \mid \ell \mid \widehat{R}_A \cdot \widehat{R}_A \mid \widehat{R}_A^* \mid \complement \widehat{R}_A \mid \widehat{R}_A + \widehat{R}_A \mid \widehat{R}_A \& \widehat{R}_A$$

Based on the concretization function  $\gamma$  we recall in Fig. 4 the denotation of LREs. Note how the concretization of a single  $\ell \in A$  results in a set of one-element strings over C. Our assumptions have a number of consequences: in particular the abstract domain A inherits structure of  $\wp(C)$ : it is also a complete, atomic, and atomistic lattice. In addition it follows that  $\gamma$  is strict,  $\alpha : Atoms(\wp(C)) \longrightarrow Atoms(A)$  is surjective, and that atoms have no overlapping meaning:  $a \neq a' \implies \gamma(a) \cap \gamma(a') = \emptyset$  (Midtgaard et al. 2016). As a consequence  $\langle \wp(C); \subseteq \rangle \xleftarrow{\gamma}{\alpha} \langle A; \sqsubseteq \rangle$  is an atomistic Galois insertion.

LREs are ordered by the language inclusion ordering:  $r \sqsubset r' \iff \mathcal{L}(r) \subseteq \mathcal{L}(r')$ . This ordering however only constitutes a pre-order, as it is not anti-symmetric: e.g.,  $\emptyset$  and odd & even are ordered both ways (their denotation is  $\emptyset$ ) but they are not syntactically identical:  $\emptyset \neq odd \& even$ . For this reason we can consider LREs up to language equivalence to regain a partial order,  $\widehat{R}_A/\approx$ . The resulting quotient set constitutes a lattice structure, 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.

As a fundamental operation over the LREs, we consider the Brzozowski derivative (Brzozowski 1964) defined in Fig. 5. Just as

$$\begin{split} \widehat{\mathcal{D}}_{a}(\emptyset) &= \emptyset \\ \widehat{\mathcal{D}}_{a}(\epsilon) &= \emptyset \\ \widehat{\mathcal{D}}_{a}(\ell) &= \begin{cases} \epsilon & a \sqsubseteq \ell \\ \emptyset & a \not\sqsubseteq \ell \end{cases} \\ \widehat{\mathcal{D}}_{a}(r^{*}) &= \widehat{\mathcal{D}}_{a}(r) \cdot r^{*} \\ \widehat{\mathcal{D}}_{a}(r_{1} \cdot r_{2}) &= \begin{cases} \widehat{\mathcal{D}}_{a}(r_{1}) \cdot r_{2} + \widehat{\mathcal{D}}_{a}(r_{2}) & \epsilon \sqsubseteq r_{1} \\ \widehat{\mathcal{D}}_{a}(r_{1}) \cdot r_{2} & \epsilon \not\boxtimes r_{1} \end{cases} \\ \widehat{\mathcal{D}}_{a}(\complement) \cdot r_{2} & \epsilon \not\boxtimes r_{1} \end{cases} \\ \widehat{\mathcal{D}}_{a}(\complement) &= \complement \widehat{\mathcal{D}}_{a}(r) \\ \widehat{\mathcal{D}}_{a}(r_{1} + r_{2}) &= \widehat{\mathcal{D}}_{a}(r_{1}) + \widehat{\mathcal{D}}_{a}(r_{2}) \\ \widehat{\mathcal{D}}_{a}(r_{1} \& r_{2}) &= \widehat{\mathcal{D}}_{a}(r_{1}) \& \widehat{\mathcal{D}}_{a}(r_{2}) \end{split}$$

Figure 5: Brzozowski derivative  $\widehat{\mathcal{D}} : \widehat{R}_A \longrightarrow Atoms(A) \longrightarrow \widehat{R}_A$  of LREs

$$\begin{split} \widehat{range}(\emptyset) &= to\_equivs(\top) \\ \widehat{range}(\epsilon) &= to\_equivs(\top) \\ \widehat{range}(\ell) &= to\_equivs(\top) \\ \widehat{range}(\ell) &= to\_equivs(\ell) \\ \widehat{range}(r^*) &= \widehat{range}(r) \\ \widehat{range}(r_1 \cdot r_2) &= \begin{cases} \widehat{overlay}(\widehat{range}(r_1), \widehat{range}(r_2)) & \epsilon \sqsubseteq r_1 \\ \widehat{range}(r_1) & \epsilon \trianglerighteq r_1 \\ \widehat{range}(r_1 + r_2) &= \widehat{overlay}(\widehat{range}(r_1), \widehat{range}(r_2)) \\ \widehat{ange}(r_1 \& r_2) &= \widehat{overlay}(\widehat{range}(r_1), \widehat{range}(r_2)) \end{split}$$

Figure 6: Partition function  $\widehat{range} : \widehat{R}_A \longrightarrow \widehat{equiv}_A$  of LREs

in a plain Brzozowski derivative, the lattice-valued generalization computes a new LRE, representing the language that remains after having matched an atom *a* as the first character:

Lemma 4.1 (Meaning of derivatives (Midtgaard et al. 2016)).

$$\forall a \in Atoms(A), r \in R_A.$$
$$\mathcal{L}(\widehat{\mathcal{D}}_a(r)) = \{ w \mid \forall c \in \gamma(a). \ cw \in \mathcal{L}(r) \}$$

### 4.2 Partitioning atoms

 $\tilde{r}$ 

 $\tilde{\tau}$ 

A partition P of a set S is a non-empty set of subsets  $P \subset \wp(S)$ , such that (1) the equivalence classes (or blocks) are non-empty  $\forall X \in P. \ X \neq \emptyset$ , (2) any two equivalence classes have nothing in common  $(X, Y \in P. \ X \neq Y \implies X \cap Y = \emptyset)$ , and (3) the equivalence classes collectively span all of S ( $\cup P = S$ ). We formulate in Fig. 6 a generic partition function  $\widehat{range}$ , that for a given  $r \in \widehat{R}_A$  partitions Atoms(A) into equivalence classes such that two atoms a, a' belong to the same equivalence class if  $\widehat{\mathcal{D}}_a(r) = \widehat{\mathcal{D}}_{a'}(r)$  and write  $\widehat{equiv}_A$  for the type of atom partitions. The partition function  $\widehat{range}$  is in turn based on the ability to partition A's atoms, which we phrase in terms of two primitives

$$\begin{array}{c} to\_equivs: A \longrightarrow equiv_A \\ \widehat{overlay}: \widehat{equiv}_A \longrightarrow \widehat{equiv}_A \longrightarrow \widehat{equiv}_A \end{array}$$

of which the former should return a partition of a given latticevalued literal  $\ell \in A$  and the latter should refine two partitions into a third partition, finer than both its arguments. Partitions form a lattice ordered by refinement (Grätzer 1978). As such,  $to\_equivs(\top)$  returns the partition that identifies all atoms (top of the partition lattice) and *overlay* computes a lower bound (an under-approximation of the greatest lower bound). We can thus instantiate  $\overline{range}$  to partition, e.g., interval-valued regular expressions or parity-valued regular expressions by a suitable parameterization of  $to\_equivs$  and  $\overline{overlay}$ .

Finally we require two additional operations

$$\widehat{repr} : (\wp(Atoms(A)) \setminus \{\emptyset\}) \longrightarrow Atoms(A)$$
$$\widehat{project} : (\wp(Atoms(A)) \setminus \{\emptyset\}) \longrightarrow A$$

which both accept an equivalence class [a] from a partition of Atoms(A). Specifically we require that  $\widehat{repr}([a'])$  returns an element  $a \in Atoms(A)$  in its argument equivalence class  $(a \in [a'])$  and we require that  $\widehat{project}([a'])$  returns an element in A that accounts for all atoms a in its argument equivalence class:  $\forall a \in [a']$ .  $a \sqsubseteq \widehat{project}([a'])$  and that  $\widehat{project}$  is monotone.

**Partitioning Interval Atoms** Intervals over integers  $Interval = \{[l; u] \mid l \le u \land l \in \mathbb{Z} \cup \{-\infty\} \land u \in \mathbb{Z} \cup \{+\infty\}\} \cup \{\bot\}$  form a complete lattice when ordered under interval inclusion (Cousot and Cousot 1976) with a Galois insertion:

$$\begin{split} \langle \wp(\mathbb{Z}); \subseteq \rangle & \xleftarrow{\gamma_{Int}} \langle Interval; \sqsubseteq \rangle & \text{where} \\ \alpha_{Int}(S) = \begin{cases} \bot & S = \emptyset & \gamma_{Int}(\bot) = \emptyset \\ [\inf S; \sup S] & S \neq \emptyset & \gamma_{Int}([l;u]) = \{i \mid l \leq i \leq u\} \end{cases} \end{split}$$

In this domain the atoms are the unit intervals:  $Atoms(Interval) = \{[i; i] \mid i \in \mathbb{Z}\}$ . We can furthermore partition the atoms (the unit intervals) by representing the equivalence classes as a set of non-overlapping, non-empty intervals that collectively span the entire domain:  $\widehat{equiv}_{Interval} = \wp(Interval \setminus \{\bot\})$ .

Fig. 7 defines to equivs and overlay for the interval domain. For example, to equivs ([1;5]) returns a partition consisting of three equivalence classes  $\{[-\infty; 0], [1;5], [6; +\infty]\}$ . We can combine this partition with another partition  $\{[-\infty; 2], [3; +\infty]\}$  using overlay, such that the result  $\{[-\infty; 0], [1; 2], [3; 5], [6; +\infty]\}$  is a partition which is a refinement of both arguments. For an equivalence class (represented as a non-bottom interval) we can now let  $\widehat{repr}$  return a unit interval less than its argument, e.g.,  $\widehat{repr}([6; +\infty]) = [6; 6]$ , whereas  $\widehat{project}$  can be defined as the identity function  $\widehat{project}([l; u]) = [l; u]$  since non-bottom interval domain. The latter definition is clearly monotone over the sets-of-atoms and interval orderings.

**Partitioning Cartesian Product Atoms** In static analysis there are multiple ways to form product lattices, e.g., as Cartesian products  $L_1 \times L_2$  ordered componentwise. For abstractions defined by a Galois insertion with a strict  $\gamma$ , this implicitly means that the atoms of  $L_1 \times L_2$  are of the shape  $\langle \ell_1, \bot \rangle \in Atoms(L_1) \times L_2$  and  $\langle \bot, \ell_2 \rangle \in L_1 \times Atoms(L_2)$ . Hence we can partition  $Atoms(L_1 \times L_2)$  with a pair  $\widehat{equiv}_{L_1} \times \widehat{equiv}_{L_2}$  where the first component partitions  $\{\bot\} \times Atoms(L_2)$  and the second component partitions  $Atoms(L_1) \times \{\bot\}$ . As a consequence we can compositionally write

$$\begin{split} & to\_equivs(\ell_1,\ell_2) = (to\_equivs_{L_1}(\ell_1), to\_equivs_{L_2}(\ell_2)) \\ \widehat{verlay}((P_1,P_2), (P_1',P_2')) = (\widehat{overlay}_{L_1}(P_1,P_1'), \widehat{overlay}_{L_2}(P_2,P_2')) \end{split}$$

where  $P_1, P'_1$  range over partitions of  $Atoms(L_1)$  and  $P_2, P'_2$ range over partitions of  $Atoms(L_2)$ . Given a single equivalence class in  $(Atoms(L_1) \times \{\bot\}) \cup (\{\bot\} \times Atoms(L_2))$  it will belong to either of the two sets of the disjoint union. The definitions of

0

$$\widehat{to\_equivs}([l;u]) = \begin{cases} [-\infty; +\infty] & l = -\infty \land u = +\infty \\ [-\infty; u], [u+1; +\infty] & l = -\infty \land u \neq +\infty \\ [-\infty; l-1], [l; +\infty] & l \neq -\infty \land u = +\infty \\ [-\infty; l-1], [l;u], [u+1; +\infty] & l \neq -\infty \land u = +\infty \\ [-\infty; l-1], [l;u], [u+1; +\infty] & l \neq -\infty \land u \neq +\infty \end{cases}$$

$$\widehat{overlay}([l_1; +\infty], [l_2; +\infty]) = [l_1; +\infty] & l_1 = l_2 \text{ holds as an invariant}$$

$$\widehat{overlay}([l_1; u_1] :: R'_1, [l_2; u_2] :: R'_2) = \begin{cases} [l_1; u_1] :: \widehat{overlay}(R'_1, R'_2) & l_1 = l_2 \land u_1 = u_2 \\ [l_1; u_1] :: \widehat{overlay}(R'_1, [u_1 + 1; u_2] :: R'_2) & l_1 = l_2 \land u_1 < u_2 \\ [l_2; u_2] :: \widehat{overlay}([u_2 + 1; u_1] :: R'_1, R'_2) & l_1 = l_2 \land u_1 > u_2 \end{cases}$$

Figure 7: to equivs and overlay for the interval domain

 $\widehat{repr}$  and *project* therefore both dispatch:

$$\begin{split} \widehat{repr}([a]_1) &= (\widehat{repr}_{L_1}([a]), \bot) \quad \widehat{project}([a]_1) &= (\widehat{project}_{L_1}([a]), \bot) \\ \widehat{repr}([a]_2) &= (\bot, \widehat{repr}_{L_2}([a])) \quad \widehat{project}([a]_2) &= (\bot, \widehat{project}_{L_2}([a])) \end{split}$$

Again this definition of project is monotone under the sets-ofatoms and componentwise ordering when composed from monotone definitions of project for  $L_1$  and  $L_2$ .

**Partitioning Reduced/Smash Product Atoms** An alternative form of lattice product often found in static analysis is the reduced or smash products  $L_1 * L_2 = ((L_1 \setminus \{\bot\}) \times (L_2 \setminus \{\bot\})) \cup \{\bot\}$  again ordered componentwise with the added  $\bot$  being less than all other elements. For this domain the atoms are a combination of atoms from each of  $L_1$  and  $L_2$ :  $Atoms(L_1 * L_2) = Atoms(L_1) \times Atoms(L_2)$ . There are several ways to partition this set of atoms. One is by using a pair of partitions  $\widehat{equiv}_{L_1} \times \widehat{equiv}_{L_2}$ . We define  $to\_equivs_{L_1 * L_2}$  and  $\widehat{overlay}_{L_1 * L_2}$  as in Cartesian products. Similarly  $\widehat{repr}$  and  $\widehat{project}$  are defined compositionally:

$$\widehat{repr}([a_1], [a_2]) = (\widehat{repr}_{L_1}([a_1]), \widehat{repr}_{L_2}([a_2]))$$

$$\widehat{project}([a_1], [a_2]) = (\widehat{project}_{L_1}([a_1]), \widehat{project}_{L_2}([a_2]))$$

Again the above definition of *project* is monotone under the sets-of-atoms and componentwise ordering when composed from monotone definitions of *project* for  $L_1$  and  $L_2$ .

Alternatively one can partition  $Atoms(L_1 * L_2)$  using a function space:  $(\wp(Atoms(L_1)) \setminus \{\emptyset\}) \longrightarrow \widehat{equiv}_{L_2}$  (or vice versa). Such a representation allows us to partition pairs with, e.g., first component [1; 5] independently of the other first components, e.g.,  $[-\infty; 0]$  and  $[6; +\infty]$ , thereby potentially leading to a coarser product partition.

With the LREs in place, we now turn to the static analysis.

# 4.3 The analysis proper

The basic idea of the analysis is to extend a base analysis (that approximates stores) for the core imperative language with two additional components: futures and histories. Futures (ranged over by  $\hat{f}$ ) are *consumed* by the analysis and describe what network communication the environment offers, whereas histories (ranged over by  $\hat{h}$ ) are *produced* by the analysis as an instrumented (approximate) trace of network communication of the process in question. Both futures and histories are expressed as LREs.

The analysis works for a generic value domain Val that is

- defined as a Galois insertion  $\langle \wp(Val); \subseteq \rangle \xleftarrow{\gamma_v}{\alpha_v} \langle \widehat{Val}; \sqsubseteq \rangle$ , with a strict  $\gamma_v$
- · connecting complete lattices, and
- with sound, monotone over-approximations of the arithmetic operators:  $\hat{+}$  of + and  $\hat{-}$  of -.

These requirements are easily satisfied by a range of standard lattices, such as parity, sign, constant propagation, and interval. To approximate a set of stores (partial functions  $Var \rightarrow Val$ ) the analysis uses a lifted function space  $\widehat{Store} = (Var \rightarrow \widehat{Val})_{\perp}$ and we let  $\widehat{\rho}$  range over these. The lifting lets us distinguish the empty store  $\lambda x. \perp_v$  from an unreachable program point  $\perp$ . Assuming channels have been numbered (for simplicity in the formalism we assume ch is a number), the analysis uses two reduced product domains  $Interval * \widehat{Val}$  for characterizing reads and writes, respectively. Their Cartesian product then captures both reads and writes:  $\widehat{Ch}(\widehat{Val}) = (Interval * \widehat{Val}) \times (Interval * \widehat{Val})$ . The futures and histories now range over  $\widehat{Ch}(\widehat{Val})$ -valued regular expressions.

To ease the manipulation of labeled statements we make use of two auxiliary functions defined in Fig. 8: *first* for fetching the label of the first statement of a potentially composite statement and *last* for fetching the labels of the last statements of a potentially composite statement. The alert reader may wonder why *first* just returns a single label in the presence of the non-deterministic  $\oplus$ construct. The reason is that the label  $\ell$  in  $s_1 \oplus^{\ell} s_2$  represents the first, implicit action (a non-deterministic choice) of either executing the left or the right branch, corresponding to a branch-node in a flow graph. On the other hand, the last statement executed of, e.g.,  $s_1 \oplus^{\ell} s_2$  may be in either  $s_1$  or  $s_2$ , hence *last* returns a set of labels. In addition the analysis utilizes the following auxiliary functions:

- $\widehat{\mathcal{A}}: E \longrightarrow \widehat{Store} \longrightarrow \widehat{Val}$  for analyzing arithmetic expressions,
- assign : Store × Var × Val → Store for modeling assignment over abstract stores, and
- two 'filter functions'

$$\widehat{true} : B \longrightarrow \widehat{Store} \longrightarrow \widehat{Store}$$
$$\widehat{false} : B \longrightarrow \widehat{Store} \longrightarrow \widehat{Store}$$

for picking up precision from comparisons in Boolean expressions.

The definition of  $\hat{\mathcal{A}}$  in Fig. 8 is generic—it works for any value abstraction satisfying the above requirements. Similarly Fig.9 presents a generic definition of filter functions  $\widehat{true}$  and  $\widehat{false}$ . These definitions will work equally well over, e.g., the parity lattice and the interval lattice, but they will not allow the analysis to pick up additional information from conditions in if-statements and while-loops. Alternatively Fig. 10 phrases domain-specific definitions of  $\widehat{true}$  and  $\widehat{false}$  for the interval lattice. For example, if we know that  $x \in [3; 10]$  and  $y \in [2; 8]$  in an abstract store  $\hat{\rho}$ ,  $\widehat{true}(x < y, \hat{\rho}) = \hat{\rho}[x \mapsto [3; 7], y \mapsto [4; 8]]$  which models that x can be at most 7 and that y will be at least 4 for the comparison x < y to evaluate to *true*.

The analysis is formulated in Fig. 11 as a syntax-directed specification over labeled statements that associates to each label  $\ell$  an

Figure 8: Auxiliary functions first, last,  $\widehat{A}$ , and  $\widehat{assign}$ 

$$\begin{split} \widehat{true}(\texttt{tt}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{tt}, \widehat{\rho}) = \bot \\ \widehat{true}(\texttt{ff}, \widehat{\rho}) &= \bot & \widehat{false}(\texttt{tt}, \widehat{\rho}) = \bot \\ \widehat{false}(\texttt{tt}, \widehat{\rho}) &= \bot & \widehat{false}(\texttt{tt}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} & \widehat{\rho} \\ \widehat{false}(\texttt{ff}, \widehat{\rho}) &= \widehat{\rho} \\ \widehat{\rho}(\texttt{fi}, \widehat{\rho}) &= \widehat{\rho} \\ \widehat{\rho}(\texttt{fi$$

Figure 10: Definitions of  $\widehat{true}$  and  $\widehat{false}$  for the interval lattice

$$\begin{split} \widehat{true}(\texttt{tt}, \widehat{\rho}) &= \widehat{false}(\texttt{ff}, \widehat{\rho}) = \widehat{\rho} \\ \widehat{true}(\texttt{ff}, \widehat{\rho}) &= \widehat{false}(\texttt{tt}, \widehat{\rho}) = \bot \\ \widehat{true}(x_1 < x_2, \widehat{\rho}) &= \widehat{false}(x_1 < x_2, \widehat{\rho}) \\ &= \begin{cases} \bot & \text{if } \widehat{\rho} = \bot \lor \widehat{\rho}(x_1) = \bot \lor \widehat{\rho}(x_2) = \bot \\ \widehat{\rho} & \text{otherwise} \end{cases} \end{split}$$

Figure 9: Generic definitions of  $\widehat{true}$  and  $\widehat{false}$ 

abstract store, a history, and a future collected in a triple  $(\hat{\rho}, \hat{h}, \hat{f})$ . The specification uses two maps  $\hat{\mathcal{E}}$  (for *entry*) and  $\hat{\mathcal{X}}$  (for *exit*) to associate such a triple as a precondition available in  $\hat{\mathcal{E}}(\ell)$  and as a postcondition available in  $\hat{\mathcal{X}}(\ell)$ . For the rest of the article we will sometimes use the shorthand notation  $\hat{\mathcal{E}}_{\rho}(\ell), \hat{\mathcal{E}}_{h}(\ell), \hat{\mathcal{E}}_{f}(\ell)$  for the projection of each of the three components  $(\hat{\rho}, \hat{h}, \hat{f})$  of  $\hat{\mathcal{E}}(\ell)$  and similarly for  $\hat{\mathcal{X}}$ .

With the above auxiliary functions in mind the cases for skip, assignment, sequencing, conditionals, and while-loops are relatively standard. For example, the sequencing rule passes the output (the postcondition) of the *last* statement of  $s_1$  to the input (the precondition) of the *first* statement of  $s_2$ . Similarly, the conditional rule passes the abstract stores flowing into  $\hat{\mathcal{E}}(\ell)$  through the filter functions  $\widehat{true}$  and  $\widehat{false}$ , thereby limiting the flow of stores into each branch. The rule for non-deterministic choice simply propagates flow from the entry (the precondition) to each branch and joins them together at the exit (the postcondition). The rule for stop<sup> $\ell$ </sup> captures that no stores can occur after halting. Since none of these constructs involve network communication,  $\hat{h}$  and  $\hat{f}$  are passed around unmodified.

In the two cases for network communication we use the shorthand notation  $[ch!v_a]$  and  $[ch?v_a]$  to denote equivalence classes  $[\langle(\bot,\bot),([ch;ch],[v_a;v_a])\rangle]$  and  $[\langle([ch;ch],[v_a;v_a]),(\bot,\bot)\rangle]$ over atom writes and atom reads in  $\widehat{Ch}(\widehat{Val})$ , respectively. In the read rule we first utilize  $\widehat{range}$  to compute a partition into equivalence classes. For each equivalence class we then

- use a projection of the equivalence class to obtain an approximation of the values  $\hat{v}$  that can flow into the store at entry x,
- record in the history that a network read took place, and
- use a representative to derive a new future from the old  $\hat{f}$ .

The write rule is similar, except we only propagate flow when the abstract value  $\hat{v}'$  being written and the abstract value expected to be read by the environment  $\hat{v}$  may have concrete values in common (a non-bottom meet).

The basic analysis given Fig. 11, analyzes a single process against a given future  $\hat{f}$ . As such, it works for analyzing individual processes against a network environment policy given by  $\hat{f}$ . We will then in Sec. 5 develop algorithms that iterate this 'intra-process analysis' to analyze two-process programs.

# 5. The Analysis Algorithm

In this section we consider the algorithmic aspects of computing analysis solutions. In particular we consider widening operators required to ensure termination, we describe an intra-process analysis algorithm, we develop two analysis algorithms that iterate the intra-process analysis to propagate analysis information between

$$\begin{split} \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\lambda}} &\models \operatorname{skip}^{\ell} \text{ iff } \widehat{\boldsymbol{\varepsilon}}(\ell) \sqsubseteq \widehat{\boldsymbol{\chi}}(\ell) \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\lambda}} &\models x :=^{\ell} e \text{ iff } (\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(e, \widehat{\rho})), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\boldsymbol{\chi}}(\ell) \text{ where } (\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\boldsymbol{\varepsilon}}(\ell) \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models x_{1} ; s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \forall \ell_{1} \in \operatorname{last}(s_{1}), \widehat{\boldsymbol{\chi}}(\ell_{1}) \sqsubseteq \widehat{\boldsymbol{\varepsilon}}(first(s_{2})) \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models if b^{\ell} \text{ then } s_{1} \text{ else } s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \\ & (\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\boldsymbol{\varepsilon}}(first(s_{1})) \land (\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\boldsymbol{\varepsilon}}(first(s_{2})) \land \\ & \forall \ell_{1} \in \operatorname{last}(s_{1}), \widehat{\boldsymbol{\chi}}(\ell_{1}) \sqsubseteq \widehat{\boldsymbol{\chi}}(\ell) \land \forall \ell_{2} \in \operatorname{last}(s_{2}), \widehat{\boldsymbol{\chi}}(\ell_{2}) \sqsubseteq \widehat{\boldsymbol{\chi}}(\ell) \\ & \text{where } (\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\boldsymbol{\varepsilon}}(\ell) \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models \text{while } b^{\ell} \text{ do } s_{1} \text{ end iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}} \\ & (\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\boldsymbol{\varepsilon}}(first(s_{1})) \land (\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\boldsymbol{\chi}}(\ell) \land \\ & \forall \ell_{1} \in \operatorname{last}(s_{1}), \widehat{\boldsymbol{\chi}}(\ell_{1}) \sqsubseteq \widehat{\boldsymbol{\xi}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models \text{while } b^{\ell} \text{ do } s_{1} \text{ end iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) &\sqsubseteq \widehat{\boldsymbol{\varepsilon}(first(s_{1})) \land (\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) &\sqsubseteq \widehat{\boldsymbol{\varepsilon}}(\ell) \land \\ & \forall \ell_{1} \in \operatorname{last}(s_{1}), \widehat{\boldsymbol{\chi}}(\ell_{1}) &\sqsubseteq \widehat{\boldsymbol{\varepsilon}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) &\subseteq \widehat{\boldsymbol{\varepsilon}(first(s_{1})) \land \hat{\boldsymbol{\varepsilon}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) &\sqsubseteq \widehat{\boldsymbol{\varepsilon}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} s_{2} \text{ iff } \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) \\ \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}} &\models s_{1} \land \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) \\ & \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) \\ \\ \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\chi}} &\models s_{1} \oplus^{\ell} \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}}, \widehat{\boldsymbol{\varepsilon}} &\models s_{2} \land \widehat{\boldsymbol{\varepsilon}}(\ell) \\$$

Figure 11: Static analysis of the process language

$$rev(\emptyset) = \emptyset \qquad rev(r^*) = (rev(r))^*$$

$$rev(\epsilon) = \epsilon \qquad rev(r_1 + r_2) = rev(r_1) + rev(r_2)$$

$$rev(\ell) = \ell \qquad rev(r_1 \& r_2) = rev(r_1) \& rev(r_2)$$

$$rev(r_1 \cdot r_2) = rev(r_2) \cdot rev(r_1) \qquad rev(\complement r) = \complement rev(r)$$

Figure 12: Symbolic reversal  $rev : \widehat{R}_A \longrightarrow \widehat{R}_A$  of LREs

the involved processes, and we discuss a prototype implementation realizing these.

# 5.1 Widening

Kleene iteration of a monotone function over LREs is not guaranteed to terminate since the domain does not satisfy the ACC. As a consequence we need a widening operator. In Midtgaard et al. (2016) we proposed an initial widening operator  $\nabla$  for LREs. By utilizing that regular languages are closed under reversal, in this section we phrase an alternative widening operator that works by reversing both its arguments, widening with  $\nabla$ , and finally reversing the result.

We first phrase reversal as a symbolic operation over LREs in Fig. 12. To prove that the approach constitutes a widening operator we need a few helper lemmas, starting with two reversals yielding the identity.

**Lemma 5.1** (*rev* twice is identity).  $\forall r \in \widehat{R}_A$ . rev(rev(r)) = r

Secondly, we prove that the symbolic reversal actually reserves the denoted language.

**Lemma 5.2** (rev is reverse).  $\forall r \in \widehat{R}_A$ .  $\mathcal{L}(rev(r)) = rev(\mathcal{L}(r))$ where  $rev(S) = \{rev(s) \mid s \in S\}$  Finally, we need that rev is monotone with respect to the language ordering.

**Lemma 5.3** (rev is monotone).  $\forall r, r' \in \widehat{R}_A. r \sqsubset r' \implies rev(r) \sqsubseteq rev(r')$ 

We are now in position to prove the main result: that 'reverse widening' constitutes a widening.

**Theorem 5.1** (Reversing a widening operator over LREs). If  $\nabla$  is a widening operator then  $r \nabla_{rev} r' = rev(rev(r) \nabla rev(r'))$  is a widening operator.

### 5.2 Intra-process analysis algorithm

We can perform an intra-process analysis of one process given an LRE describing the communication behaviour of the other process. The result of such an intra-process analysis represents a post-fixed point corresponding to the analysis specification of Fig. 11. To ensure termination of the intra-process analysis we compute the solution by iteration with widening and a subsequent iteration with narrowing. Following standard abstract interpretation literature (Cousot 1981; Bourdoncle 1993), we perform a minimal amount of widening on loop headers to limit the potential information loss.

There can be more than one cause of infinite chains. If the value domain does not satisfy the ACC, such as the intervals, we need to perform widening over them to ensure termination. For example, for the interval domain we can use the standard widening operator (Cousot and Cousot 1976). We can subsequently lift the value domain's widening operator  $\nabla_v$  to form a widening operator

over abstract stores as follows:

$$\widehat{
ho} 
abla_{st} \, \widehat{
ho}' = egin{cases} \widehat{
ho}' & \widehat{
ho} = oldsymbol{eta} \ \widehat{
ho} & \widehat{
ho}' = oldsymbol{eta} \ \widehat{
ho} & \widehat{
ho}' = oldsymbol{eta} \ \lambda x. \, \widehat{
ho}(x) \, 
abla_v \, \widehat{
ho}'(x) \end{cases}$$

Finally we combine these widening operators into a widening operator for the analysis triples:

$$(\widehat{\rho},\widehat{h},\widehat{f}) \, \triangledown(\widehat{\rho}',\widehat{h}',\widehat{f}') = (\widehat{\rho} \, \triangledown_{st} \, \widehat{\rho}',\widehat{h} \, \triangledown \, \widehat{h}',\widehat{f} + \widehat{f}')$$

thereby widening over both abstract stores (and abstract values) and LREs. Note how we do not widen over futures. As futures are *consumed* by the intra-process analysis we do not need to. In particular, since there are only finitely many derivatives of an LRE (up to associativity, commutivity, and idempotence of +), we can only form finitely many different LREs by composing two derivatives of a future into new LRE futures, thereby eliminating the need for widening (Brzozowski 1964; Midtgaard et al. 2016). In contrast histories are *produced* by the analysis and widening is therefore required to infer loop invariants for communication while ensuring termination.

To improve upon a post-fixed point computed as above, we can compute a second iteration with narrowing. To do so we can similarly lift a narrowing operator over the value domain  $\Delta_v$  to a narrowing operator over abstract stores  $\Delta_{st}$ . Finally we can lift  $\Delta_{st}$  to a narrowing operator for analysis triples:

$$(\widehat{\rho}, \widehat{h}, \widehat{f}) \,\vartriangle\, (\widehat{\rho}', \widehat{h}', \widehat{f}') = (\widehat{\rho} \,\vartriangle_{st} \,\widehat{\rho}', \widehat{h}, \widehat{f})$$

This corresponds to performing no narrowing over LREs:  $r \triangle r' = r$  which trivially satisfies the narrowing requirements. We have not investigated narrowing operators over LREs at this point.

The example programming language is structured, meaning that communication analysis of

- the empty statement and assignment should give rise to epsilon,
- message sending and receiving should give rise to output/input characters,
- sequences should give rise to concatenation,
- · conditionals and choice should give rise to sum, and
- loops should give rise to Kleene star,

which suggests a purely syntactical over-approximation without the need for fixed-point computations and widening operators. Such an approximation of communication is certainly possible and will furthermore allow us to quickly infer coarser loop invariants without iteration, e.g., with worst case assumptions of values and by disabling filter functions. Finally this is also a consequence of our chosen presentation language: for an unstructured programming language with, e.g., jumps, some form of iteration is required.

### 5.3 Inter-process analysis algorithms

Given a pair of processes  $s_1, s_2$ , if  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}_{\rho}(first(s_1))$ is a sound approximation of the initial environment and  $\epsilon \models \widehat{\mathcal{E}}_h(first(s_1))$  and  $\top^* \models \widehat{\mathcal{E}}_f(first(s_1))$  then  $\bigsqcup_{\ell} \widehat{\mathcal{E}}_h(\ell)$  captures the prefix of all possible communications with the other process  $s_2$ . Hence we can use  $\bigsqcup_{\ell} \widehat{\mathcal{E}}_h(\ell)$  from the intra-process analysis of  $s_1$  as our starting point future for analyzing  $s_2$ . Since there are only a finite number of labels in a given process one can easily compute this join—either after analysis completion, or underway by joining into one big accumulator of communication prefixes. We can then subsequently use the analysis result from  $s_2$  to reanalyze  $s_1$ . This suggests an iterative round-robin inter-process analysis approach.

### **Round-robin algorithm**

The round-robin algorithm listed in Fig. 13 on the left initially analyzes  $s_1$  under worst-case assumptions about the communication behaviour of  $s_2$  and then repeatedly reanalyzes either  $s_1$  or  $s_2$  based on the analysis result of the other.

Widening ensures that the two intra-process analysis steps terminate. We leave the termination condition for the outer interprocess analysis unspecified. For example, we can limit the number of round trip analysis iterations to a constant k, which will guarantee termination.

### **Refined algorithm**

The somewhat arbitrary choice of initially analyzing the first syntactically-occurring process may affect the outcome of the analysis, if we, e.g., stick to a fixed number of round trips. To prevent this from happening we may instead analyze both processes simultaneously. The algorithm listed in Fig. 13 on the right performs such a simultaneous analysis. In effect, this refined analysis algorithm thereby requires more loop iterations to propagate information from  $s_1$  through to  $s_2$  and back to  $s_1$ .

### 5.4 Implementation and experiments

We have implemented a prototype of the first iterative analysis algorithm in OCaml and experimented with both the initial widening operator (Midtgaard et al. 2016) and well as the reverse widening described in Sec. 5.1. The prototype analysis implementation by default uses an interval domain to approximate values. It supports a slightly larger language than the minimal one described in Sec. 3, e.g., by allowing more comparison operators and by allowing comparisons between arbitrary arithmetic expressions rather than just variables. By default our prototype implementation attempts to run 5 iterations of the iterated algorithm. However it stops earlier if an iteration does not give rise to a more precise collective communication prefix. To partition atoms of the reduced/smash product *Interval* \* Val for capturing network reads and writes the implementation furthermore uses the functional representation discussed in Sec. 4.2.

The prototype is available for download at https://github. com/jmid/iterated. Furthermore we have compiled the analysis code to JavaScript using the js\_of\_ocaml compiler and hooked the resulting JavaScript code up to a web-interface. The interested reader is invited to try the (purely client-side) analysis interface at https://jmid.github.io/iterated/.

*Example 1: Round-Trip Communication* As a first example, consider the following program:

spawn	proc1()	{	ch?x }	•
spawn	proc2()	{	ch!42	]

If we first analyze proc1 under the worst case assumption  $\hat{f}_1 = \top^*$ for the future (environment) communication, we infer the collective communication prefix  $\hat{f}_2 = \epsilon + ch?[-\infty; +\infty]$ , representing that any integer value can be read from channel ch (along with the prefix  $\epsilon$ ). Using  $\hat{f}_2$  for the subsequent analysis of proc2, we infer the collective communication prefix  $\hat{f}_1 = \epsilon + ch![42; 42]$ , representing that the process will write the value 42 to channel ch (suitably prefix closed). This concludes the first iteration of the iterative analysis. In the second iteration, when we reanalyze proc1 under this more precise assumption, we infer the collective communication prefix  $\hat{f}_2 = \epsilon + ch?[42; 42]$ , representing that the process can read the value 42 from channel ch. Reanalyzing proc2 under this more precise assumption, we again infer the collective communication prefix  $\hat{f}_1 = \epsilon + ch![42; 42]$ . The analysis implementation therefore stops early after two iterations.  $\begin{array}{ll} \operatorname{proc Round-Robin-Analysis }(s_1,s_2) & \operatorname{proc Simultaneous-Analysis }(s_1,s_2) \\ \widehat{f}_1 := \top^* & \\ \operatorname{while (termination-condition)} \\ \widehat{\mathcal{E}}, \widehat{\mathcal{X}} := \text{ analyze } s_1 \operatorname{from } \epsilon \sqsubseteq \widehat{\mathcal{E}}_h(first(s_1)) \text{ and } \widehat{f}_1 \sqsubseteq \widehat{\mathcal{E}}_f(first(s_1)) \\ \widehat{f}_2 := \bigsqcup_{\ell} \widehat{\mathcal{E}}_h(\ell) & \\ \widehat{\mathcal{E}}', \widehat{\mathcal{X}}' := \text{ analyze } s_2 \operatorname{from } \epsilon \sqsubseteq \widehat{\mathcal{E}}_h'(first(s_2)) \text{ and } \widehat{f}_2 \sqsubseteq \widehat{\mathcal{E}}_f'(first(s_2)) \\ \widehat{f}_1 := \bigsqcup_{\ell} \widehat{\mathcal{E}}_h'(\ell) & \\ \widehat{f}_1 := \bigsqcup_{\ell} \widehat{\mathcal{E}}_h'(\ell) & \\ \widehat{f}_1 := \bigsqcup_{\ell} \widehat{\mathcal{E}}_h'(\ell) & \\ \end{array}$ 



*Example 2: Deadlock* As a second example, consider the following program:

spawn	proc1()	{	ch_a?x;	
			ch_b!1;	}
spawn	proc2()	{	ch_b?y;	
			ch_a!2;	}

When we first analyze proc1 under the worst case assumption  $\widehat{f}_1 = \top^*$ , we infer the collective communication prefix  $\widehat{f}_2 =$  $\epsilon + \mathtt{ch\_a}?[-\infty; +\infty] + (\mathtt{ch\_a}?[-\infty; +\infty] \cdot \mathtt{ch\_b}![1;1]), \text{ representation}$ senting that any integer value can first be read from channel ch\_a followed by an output of value 1 on channel ch\_b. Using  $f_2$  for the subsequent analysis of proc2, we infer the collective communication prefix  $f_1 = \epsilon$ , representing that the process cannot perform any successful network communication with the given environment. This situation arises from the read-case which will perform a derivative of  $\hat{f}_2$  with respect to a representative write atom ch\_b! $\hat{v}_a$ , thereby resulting in the empty future  $\emptyset$ . We therefore have no other contributions to the collective communication prefix than the initial  $\epsilon$  history. This step furthermore concludes the first iteration. In the second iteration, when we reanalyze proc1 under this more precise assumption, we infer the collective communication prefix  $\hat{f}_2 = \epsilon$ , representing that the first process similarly cannot perform any successful network communication. Reanalyzing proc2 under this more precise assumption, we again infer the collective communication prefix  $f_1 = \epsilon$ . Again the analysis implementation therefore stops early after two iterations.

Example 3: Non-termination As a third example, consider the program in Fig. 1 from the introduction. When we first analyze proc1 under the worst case assumption we infer the collective communication prefix  $\hat{f}_2 = \epsilon + ch?[-\infty; +\infty] \cdot (ch?[-\infty; +\infty])^*$ which we can phrase more compactly as  $(ch?[-\infty; +\infty])^*$ . Using  $\hat{f}_2$  for the analysis of proc2, we infer the collective communication prefix  $\widehat{f}_1 = \epsilon + ((ch![1;999] + ch![1000;1000]) \cdot (ch![1;1000])^*),$ which we can also phrase more compactly as  $(ch![1; 1000])^*$ . This step concludes the first iteration. In the second iteration, when we reanalyze proc1 under this more precise assumption, we infer the collective communication prefix  $\hat{f}_2 = \epsilon + ((ch?[1;999] +$  $ch?[1000; 1000]) \cdot (ch?[1; 999] + ch?[1000; 1000])^*)$ , which we can also phrase more compactly as  $(ch?[1; 1000])^*$ . As a consequence, in the abstract store the variable x takes a value in [1; 1000]. The false filter function applied to the condition 0 < x in this abstract store yields  $\perp$  for the exit of proc1's loop and thereby proves non-termination.<sup>3</sup> After a final reanalysis of proc2 the analysis implementation again stops early.

In addition to the above examples, we have experimented with a number of examples from the literature, such as two CSP examples from Cousot and Cousot (Cousot and Cousot 1980), and a simple math server adapted from Vasconcelos, Gay, and Ravara (Vasconcelos et al. 2006)., For future work we plan to extend the supported language further. This would furthermore allow us to expose the resulting prototype to bigger examples.

# 6. Wellformedness

This section is dedicated to verifying the assumptions required for the analysis and to argue that solutions formally exist.

# 6.1 Atom-preserving Galois insertions

In order for the analysis to be well-defined over the LREs, we should ensure that the read and write abstractions satisfy the requirements of the parametric domain, i.e., that the abstractions are Galois insertions and map atoms to atoms.

**Lemma 6.1** (Read and write abstractions are atom-preserving Galois insertions).

$$\begin{split} \wp(Channel \times \{?\} \times Val) & \xleftarrow{\gamma_{rd}} \operatorname{Interval} * \widehat{Val} \\ \wp(Channel \times \{!\} \times Val) & \xleftarrow{\gamma_{wr}} \operatorname{Interval} * \widehat{Val} \\ \wp(Channel \times \{!\} \times Val) & \xleftarrow{\gamma_{wr}} \operatorname{Interval} * \widehat{Val} \\ \text{where} \quad \alpha_{rd}(S) = \bigsqcup_{ch?v \in S} (\alpha_{Int}(\{ch\}), \alpha_v(\{v\})) \\ \gamma_{rd}([l; u], \widehat{v}) = \bigcup_{\substack{ch?\gamma_{td}([l; u]) \\ v \in \gamma_v(\widehat{v})}} \{ch?v\} \\ \alpha_{wr}(S) = \bigsqcup_{ch!v \in S} (\alpha_{Int}(\{ch\}), \alpha_v(\{v\})) \\ \gamma_{wr}([l; u], \widehat{v}) = \bigcup_{\substack{ch?\gamma_{Int}([l; u]) \\ v \in \gamma_v(\widehat{v})}} \{ch!v\} \\ \sum_{\substack{ch?\gamma_{td}([l; u]) \\ v \in \gamma_v(\widehat{v})}} \{ch!v\} \\ \end{split}$$

Furthermore:

$$\begin{array}{l} \alpha_{rd} : Atoms(\wp(Channel \times \{?\} \times Val)) \longrightarrow Atoms(Interval * \widehat{Val}) \\ \alpha_{wr} : Atoms(\wp(Channel \times \{!\} \times Val)) \longrightarrow Atoms(Interval * \widehat{Val}) \end{array}$$

As a corollary the abstractions are atomistic Galois insertions.

Lemma 6.2 (Channel abstraction is an atom-preserving Galois insertion).

$$\wp(Action) \xleftarrow{\gamma_{ch}}{\alpha_{ch}} \widehat{Ch}(\widehat{Val})$$

where  $Action = (Channel \times \{?\} \times Val) \cup (Channel \times \{!\} \times Val)$   $\alpha_{ch}(S) = (\alpha_{rd}(\{ch?v \in S\}), \alpha_{wr}(\{ch!v \in S\}))$  $\gamma_{ch}(\widehat{v}_r, \widehat{v}_w) = \gamma_{wr}(\widehat{v}_w) \cup \gamma_{rd}(\widehat{v}_r)$ 

<sup>&</sup>lt;sup>3</sup> Here we utilize a comparison between a numeral 0 and a variable x, which strictly speaking is not syntactically valid for the formalized language fragment. However it is allowed by the prototype implementation. Furthermore we can achieve the same result within the bounds of the formalized frag-

ment, by initially binding the value 0 to a fresh variable zero and using a comparison zero < x instead.

Furthermore  $\alpha_{ch}$ :  $Atoms(\wp(Action)) \longrightarrow Atoms(\widehat{Ch}(\widehat{Val})).$ 

As a corollary the abstraction is an atomistic Galois insertion.

### 6.2 Monotonicity of auxiliary functions

Our auxiliary functions are monotone which we will utilize in the following subsection to argue for existence of solutions.

**Lemma 6.3** (Monotonicity of  $\widehat{A}$ ).

$$\forall e, \widehat{\rho}, \widehat{\rho}'. \ \widehat{\rho} \stackrel{:}{\sqsubseteq} \widehat{\rho} \implies \widehat{\mathcal{A}}(e, \widehat{\rho}) \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho}')$$

Lemma 6.4 (Monotonicity of assign).

$$\begin{array}{l} \forall \widehat{\rho}, \widehat{\rho}', x, \widehat{v}. \ \widehat{\rho} \sqsubseteq \widehat{\rho}' \implies \widehat{assign}(\widehat{\rho}, x, \widehat{v}) \sqsubseteq \widehat{assign}(\widehat{\rho}', x, \widehat{v}) \\ \forall \widehat{\rho}, x, \widehat{v}, \widehat{v}'. \ \widehat{v} \sqsubseteq \widehat{v}' \implies \widehat{assign}(\widehat{\rho}, x, \widehat{v}) \sqsubseteq \widehat{assign}(\widehat{\rho}, x, \widehat{v}') \end{array}$$

Lemma 6.5 (Monotonicity of  $\widehat{true}$ ,  $\widehat{false}$ ).

$$\begin{array}{l} \forall b, \widehat{\rho}, \widehat{\rho'} . \ \widehat{\rho} \stackrel{.}{\sqsubseteq} \widehat{\rho'} \implies \widehat{true}(b, \widehat{\rho}) \stackrel{.}{\sqsubseteq} \widehat{true}(b, \widehat{\rho'}) \\ & \wedge \ \widehat{\rho} \stackrel{.}{\sqsubseteq} \widehat{\rho'} \implies \widehat{false}(b, \widehat{\rho}) \stackrel{.}{\sqsubseteq} \widehat{false}(b, \widehat{\rho'}) \end{array}$$

## 6.3 Moore family failure

The denotation  $\mathcal{L}$  acts as our concretization function by mapping a LRE to the set of strings it represents. One may therefore wonder whether one can formulate a Galois connection for the LREs. We answer this question negatively: the abstract domain does not have a Galois connection (it is concretization only). In particular, we can show that  $\{\mathcal{L}(r) \mid r \in \widehat{R}_A\}$  does not constitute Moore family:

**Lemma 6.6** (Moore family failure).  $\{\mathcal{L}(r) \mid r \in \widehat{R}_A\}$  is not a Moore family

As a corollary there does not exist a Galois connection (Cousot and Cousot 1992a). However by definition  $\mathcal{L}$  preserves binary meets:  $\mathcal{L}(r_1 \& r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2).$ 

### 6.4 Existence of analysis solutions

A popular way to argue for the existence of solutions to an analysis specification such as that of Fig. 11 is to show that the solutions constitute a Moore family. However we have no hope of doing so because of our dependence on LREs. To argue that our analysis specification has solutions we instead prove that top is a valid solution and that the binary meet of two solutions is itself also a solution

Lemma 6.7 (Top is a valid analysis solution).

For all s.  $\widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s$  where  $\widehat{\mathcal{E}^{\top}} = \lambda \ell. (\lambda x. \top_A, \top^*, \top^*)$  and  $\widehat{\mathcal{X}^{\top}} = \lambda \ell. \ (\lambda x. \top_A, \top^*, \top^*)$ 

Lemma 6.8 (Meet preserves solutions).

For all  $s, \widehat{\mathcal{E}^1}, \widehat{\mathcal{X}^1}, \widehat{\mathcal{E}^2}, \widehat{\mathcal{X}^2}$ . If  $\widehat{\mathcal{E}^1}, \widehat{\mathcal{X}^1} \models s$  and  $\widehat{\mathcal{E}^2}, \widehat{\mathcal{X}^2} \models s$  then  $\widehat{\mathcal{E}^1} \sqcap \widehat{\mathcal{E}^2}, \widehat{\mathcal{X}^1} \sqcap \widehat{\mathcal{X}^2} \models s$ 

#### 7. Soundness

To build up towards a soundness proof for the entire analysis, we first establish soundness for the auxiliary functions, we then prove soundness of the instrumented semantics with respect to sequences of the original operational semantics, and finally prove soundness of the analysis with respect to the instrumented semantics.

## 7.1 Soundness of auxiliary functions

We first establish a Galois insertion connecting sets of stores to abstract stores.

**Lemma 7.1** ( $\alpha_{st}, \gamma_{st}$  is a Galois insertion).

$$\langle \wp(\operatorname{Var} \, \hookrightarrow \, \operatorname{Val}); \subseteq \rangle \xrightarrow[\alpha_{st}]{\gamma_{st}} \langle (\operatorname{Var} \, \longrightarrow \, \widehat{\operatorname{Val}})_{\perp}; \sqsubseteq \rangle$$

where

$$\begin{split} \alpha_{st}(S) &= \begin{cases} \bot & S = \emptyset \\ \lambda x. \; \alpha_v(\{\rho(x) \mid \rho \in S \; \land \; \rho(x) \text{ defined}\}) & S \neq \emptyset \end{cases} \\ \gamma_{st}(\widehat{\rho}) &= \begin{cases} \emptyset & \widehat{\rho} = \bot \\ \{\rho \mid \forall x. \; \rho(x) \text{ undefined } \lor \; \rho(x) \in \gamma_v(\widehat{\rho}(x))\} & \widehat{\rho} \neq \bot \end{cases} \end{split}$$

for some value abstraction  $\langle \wp(Val); \subseteq \rangle \xleftarrow{\gamma_v}{\alpha_v} \langle \widehat{Val}; \subseteq \rangle$  between complete lattices and where the lifted ordering  $\stackrel{.}{\sqsubseteq}$  is extended such that  $\perp$  is less or equal to all other elements.

Note that by definition  $\gamma_{st}(\widehat{\rho}) \neq \emptyset$  for  $\widehat{\rho} \neq \bot$  as the partial function undefined at all entries is always included in the result (it always fulfills the condition). Based on the Galois insertions for values and stores we can now prove soundness of  $\widehat{\mathcal{A}}$ ,  $\widehat{assign}$ ,  $\widehat{true}$ , and  $\widehat{false}$ .

Lemma 7.2 (Soundness of  $\widehat{\mathcal{A}}$ ).

$$\forall e \in E, \widehat{\rho} \in \overline{Store}. \\ \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e \Downarrow v\}) \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho})$$

Lemma 7.3 (Soundness of assign).

$$\begin{array}{l} \forall \widehat{\rho}, x, \widehat{v}. \\ \alpha_{st}(\{\rho[x \mapsto v] \mid v \in \gamma_v(\widehat{v}) \land \rho \in \gamma_{st}(\widehat{\rho})\}) \stackrel{.}{\sqsubseteq} \widehat{assign}(\widehat{\rho}, x, \widehat{v}) \end{array}$$

For the Boolean expressions we over-approximate versions of true and *false* defined as follows:

$$true(b,S) = \{ \rho \in S \mid \rho \vdash_{\mathcal{B}} b \Downarrow \texttt{tt} \}$$
$$false(b,S) = \{ \rho \in S \mid \rho \vdash_{\mathcal{B}} b \Downarrow \texttt{ff} \}$$

Lemma 7.4 (Soundness of  $\widehat{true}$  and  $\widehat{false}$ ).

$$\alpha_{st}(true(b,\gamma_{st}(\widehat{\rho}))) \stackrel{!}{\sqsubseteq} true(b,\widehat{\rho})$$
  
and  $\alpha_{st}(false(b,\gamma_{st}(\widehat{\rho}))) \stackrel{!}{\sqsubseteq} false(b,\widehat{\rho})$ 

This can be proved for both the generic definition in Fig. 9 as well as for the specialized versions such as the one in Fig. 10 which takes the details of the concrete domain, e.g., intervals, into account. In the appendix we provide the proof for both the generic and the interval versions.

### 7.2 Soundness of instrumented semantics

We formulate in Fig. 14 an instrumented operational semantics where judgments can take one of two forms, depending on whether the right-hand-side configuration is final or not:

$$\langle s^{\ell}, \rho, h \rangle \xrightarrow{\alpha} \langle s_1^{\ell_1}, \rho_1, h_1 \rangle \quad \text{or} \quad \langle s^{\ell}, \rho, h \rangle \xrightarrow{\alpha} \langle \rho_1, h_1 \rangle$$
(configurations in two forms)

The h component in a configuration records the concrete history of network actions. Even though not formulated as such, we can understand the instrumentation as an abstraction of a trace-based collecting semantics.

We can now phrase soundness of the instrumented semantics with respect to traces of the original operational semantics.

# Theorem 7.1 (Soundness of instrumented semantics).

If  $c_1 \parallel \overline{c}_1 \xrightarrow{\alpha_1,\beta_1} c_2 \parallel \overline{c}_2 \xrightarrow{\alpha_2,\beta_2} \cdots \xrightarrow{\alpha_{n-1},\beta_{n-1}} c_n \parallel \overline{c}_n$  from some initial state  $c_1 \parallel \overline{c}_1 = \langle s^{\ell}, \rho \rangle \parallel \langle \overline{s^{\ell}}, \overline{\rho} \rangle$  then there exists an

$$\frac{\overline{\langle \operatorname{skip}^{\ell}, \rho, h \rangle} \xrightarrow{\tau} \langle \rho, h \cdot \tau \rangle}{|\operatorname{ISKIP}|} \operatorname{ISKIP}} \frac{\rho \vdash_{\mathcal{A}} e \Downarrow v}{\langle x :=^{\ell} e, \rho, h \rangle \xrightarrow{\tau} \langle \rho[x \mapsto v], h \cdot \tau \rangle} \operatorname{IASSIGN}} \frac{\rho \vdash_{\mathcal{A}} e \Downarrow v}{\langle x :=^{\ell} e, \rho, h \rangle \xrightarrow{\alpha} \langle \rho[x \mapsto v], h \cdot \tau \rangle} \operatorname{ISEQ1}} \frac{\langle s_1, \rho, h \rangle \xrightarrow{\alpha} \langle s_3, s_2, \rho', h' \rangle}{\langle s_1; s_2, \rho, h \rangle \xrightarrow{\alpha} \langle s_2, \rho', h' \rangle} \operatorname{ISEQ2}} \frac{\rho \vdash_{\mathcal{B}} b \Downarrow \operatorname{tt}}{\langle \operatorname{if} b^{\ell} \operatorname{then} s_1 \operatorname{else} s_2, \rho, h \rangle \xrightarrow{\tau} \langle s_1, \rho, h \cdot \tau \rangle} \operatorname{IIF1}} \frac{\rho \vdash_{\mathcal{B}} b \Downarrow \operatorname{tt}}{\langle \operatorname{if} b^{\ell} \operatorname{then} s_1 \operatorname{else} s_2, \rho, h \rangle \xrightarrow{\tau} \langle s_2, \rho, h \cdot \tau \rangle} \operatorname{IIF2}} \frac{\rho \vdash_{\mathcal{B}} b \Downarrow \operatorname{tf}}{\langle \operatorname{if} b^{\ell} \operatorname{then} s_1 \operatorname{else} s_2, \rho, h \rangle \xrightarrow{\tau} \langle s_2, \rho, h \cdot \tau \rangle} \operatorname{IIF2}} \frac{\rho \vdash_{\mathcal{B}} b \Downarrow \operatorname{tf}}{\langle \operatorname{if} b^{\ell} \operatorname{then} s_1 \operatorname{else} s_2, \rho, h \rangle \xrightarrow{\tau} \langle s_2, \rho, h \cdot \tau \rangle} \operatorname{IIF2}}$$

 $\frac{\rho \vdash_{\mathcal{B}} b \Downarrow \mathsf{tt}}{\langle \mathsf{while} \ b^\ell \ \mathsf{do} \ s_1 \ \mathsf{end}, \rho, h \rangle \xrightarrow{\tau} \langle s_1 \ ; \mathsf{while} \ b^\ell \ \mathsf{do} \ s_1 \ \mathsf{end}, \rho, h \cdot \tau \rangle}$ 

$$\frac{\rho \vdash_{\mathcal{B}} b \Downarrow \text{if}}{\langle \text{while } b^{\ell} \text{ do } s_1 \text{ end}, \rho, h \rangle \xrightarrow{\tau} \langle \rho, h \cdot \tau \rangle} \text{ IWHILE2}$$

IWHILE1

$$\frac{}{\langle ch?^{\ell}x, \rho, h \rangle \xrightarrow{ch?v} \langle \rho[x \mapsto v], h \cdot ch?v \rangle} \text{IREAD}$$

$$\frac{\rho \vdash_{\mathcal{A}} e \Downarrow v}{\langle ch!^{\ell} e, \rho, h \rangle \xrightarrow{ch! v} \langle \rho, h \cdot ch! v \rangle} \text{IWRITE}$$

$$\frac{\langle s_1, \rho, h \rangle \xrightarrow{\alpha} ic_1}{\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle \xrightarrow{\alpha} ic_1} \text{ICHOICE1}$$

$$\frac{\langle s_2, \rho, h \rangle \xrightarrow{\alpha} ic_2}{\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle \xrightarrow{\alpha} ic_2} \text{ICHOICE2}$$

$$\frac{\langle s_1, \rho_1, h_1 \rangle \xrightarrow{\tau} ic'_1}{\langle s_1, \rho_1, h_1 \rangle \parallel ic_2 \Longrightarrow ic'_1 \parallel ic_2} \text{ ISysLeft}$$

$$\frac{\langle s_2, \rho_2, h_2 \rangle \xrightarrow{\tau} ic'_2}{ic_1 \parallel \langle s_2, \rho_2, h_2 \rangle \Longrightarrow ic_1 \parallel ic'_2} \text{ IsysRight}$$

$$\frac{\langle s_1, \rho_1, h_1 \rangle \xrightarrow{ch! v} ic'_1 \quad \langle s_2, \rho_2, h_2 \rangle \xrightarrow{ch? v} ic'_2}{\langle s_1, \rho_1, h_1 \rangle \parallel \langle s_2, \rho_2, h_2 \rangle \Longrightarrow ic'_1 \parallel ic'_2} \text{ ISysWR}$$

$$\frac{\langle s_1, \rho_1, h_1 \rangle \xrightarrow{ch?v} ic'_1 \quad \langle s_2, \rho_2, h_2 \rangle \xrightarrow{ch!v} ic'_2}{\langle s_1, \rho_1, h_1 \rangle \parallel \langle s_2, \rho_2, h_2 \rangle \Longrightarrow ic'_1 \parallel ic'_2} \text{ ISYSRW}$$

### Figure 14: Instrumented operational semantics

### instrumented trace

$$ic_{1} \parallel \overline{ic_{1}} \Longrightarrow ic_{2} \parallel \overline{ic_{2}} \Longrightarrow \dots \Longrightarrow ic_{n} \parallel \overline{ic_{n}}$$
(corr. instr. trace)
$$\land c_{i}, h_{i} = proj(ic_{i}) \land \overline{c}_{i}, \overline{h}_{i} = proj(\overline{ic}_{i}) \qquad i \in \{1, \dots, n\}$$
(with corr. confs)
$$\land \alpha_{1}\alpha_{2} \dots \alpha_{n-1} = h_{n} \land \beta_{1}\beta_{2} \dots \beta_{n-1} = \overline{h}_{n}$$
(and comm. histories)

where  $\alpha$  and  $\beta$  range over  $\epsilon$  (no computation step),  $\tau$  (no communication), ch?v (channel read), ch!v (channel write), and where  $proj: \Sigma_I \longrightarrow \Sigma$  is defined as follows:

$$proj(\langle \rho, h \rangle) = \rho, h$$
$$proj(\langle s^{\ell}, \rho, h \rangle) = \langle s^{\ell}, \rho \rangle, h$$

In the theorem we furthermore utilize that  $\epsilon$  doubles as the empty string of actions. To prove the theorem we apply the following helper lemma that relates single steps in the two semantics.

**Lemma 7.5** (Process step soundness). If  $\langle s^{\ell}, \rho \rangle, h = proj(\langle s^{\ell}, \rho, h \rangle)$  and  $\langle s^{\ell}, \rho \rangle \xrightarrow{\alpha} c$  then  $\langle s^{\ell}, \rho, h \rangle \xrightarrow{\alpha} ic$  such that  $c, (h \cdot \alpha) = proj(ic)$ 

where  $\alpha$  ranges over  $\tau$ , ch?v, and ch!v.

To phrase soundness of the analysis with respect to the instrumented semantics it will be convenient to *decouple* a system trace into two process traces:

Lemma 7.6 (Decoupling a system trace). If

$$ic_1 \parallel \overline{ic_1} \Longrightarrow ic_2 \parallel \overline{ic_2} \Longrightarrow \ldots \Longrightarrow ic_n \parallel \overline{ic_n}$$

then we can decouple the system trace into two process traces

$$ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{n-1}} ic_n$$
$$\overline{ic_1} \xrightarrow{\beta_1} \overline{ic_2} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \overline{ic_n}$$

where we write  $ic \stackrel{\epsilon}{\longrightarrow} ic'$  when ic = ic'.

Since two processes in a system trace do not necessarily perform equally many computation steps, such  $\epsilon$  process steps nevertheless lets us decouple a system trace into process traces of equal length. For example, this may happen if there are two processes each containing a loop but where the loop of the first process performs three steps for each step in the second process's loop.

### 7.3 Soundness of analysis

To prove soundness of the analysis with respect to the instrumented semantics, we first define a structural,  $\tau$ -deleting operation:

$$|\epsilon| = \epsilon \qquad |\alpha \cdot h| = \begin{cases} |h| & \alpha = \tau \\ \alpha \cdot |h| & \alpha \neq \tau \end{cases}$$

We will use |-| for a homomorphic/elementwise abstraction of communication histories. By two simple induction arguments we can prove the following lemma about |-|:

Lemma 7.7 (Actions at the end of a string).

 $|h \cdot \tau| = |h| \wedge |h \cdot \alpha| = |h| \cdot \alpha \text{ for } \alpha \neq \tau$ 

Again a little helper lemma comes handy, this one concerning *last*.

# **Lemma 7.8** (Preservation of *last*). $\forall s, s_1, \rho, \rho_1, h, h_1, \alpha. \langle s, \rho, h \rangle \xrightarrow{\alpha} \langle s_1, \rho_1, h_1 \rangle \implies last(s_1) \subseteq last(s)$

We are now ready to phrase soundness of a single process step. We express this property in two lemmas depending on whether we transition to a final configuration (Lemma 7.9) or not (Lemma 7.10). Lemma 7.9 (Local soundness of store and history specification 1).

$$\begin{split} \forall s, \rho, \rho_1, h, h_1, \alpha, \widehat{\mathcal{E}}, \widehat{\mathcal{X}}, f. \\ \langle s, \rho, h \rangle & \xrightarrow{\alpha} \langle \rho_1, h_1 \rangle \wedge \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash s \wedge \rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)) \wedge \\ |h| \in \mathcal{L}(\widehat{\mathcal{E}_h}(\ell)) \wedge |\overline{\alpha} \cdot f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(\ell)) \text{ where } \ell = first(s) \\ \implies h_1 = h \cdot \alpha \wedge \forall \ell_1 \in last(s). \ \rho_1 \in \gamma_{st}(\widehat{\mathcal{X}_{\rho}}(\ell_1)) \wedge \\ |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{X}_h}(\ell_1)) \wedge |f| \in \mathcal{L}(\widehat{\mathcal{X}_f}(\ell_1)) \end{split}$$

using the notation  $\overline{\tau} = \tau$   $\overline{ch?v} = ch!v$   $\overline{ch!v} = ch?v$ for expressing a converse network action.

**Lemma 7.10** (Local soundness of store and history specification 2).

$$\begin{split} \forall s, s_1, \rho, \rho_1, h, h_1, \alpha, \widehat{\mathcal{E}}, \widehat{\mathcal{X}}, f. \\ \langle s, \rho, h \rangle & \stackrel{\alpha}{\longrightarrow} \langle s_1, \rho_1, h_1 \rangle \land \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash s \land \rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)) \land \\ |h| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell)) \land |\overline{\alpha} \cdot f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell)) \text{ where } \ell = first(s) \\ & \implies \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash s_1 \land h_1 = h \cdot \alpha \land \rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell_1)) \land \\ |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell_1)) \land |f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell_1)) \\ & \text{ where } \ell_1 = first(s_1) \end{split}$$

We can now utilize the local single process soundness results to prove soundness of a system consisting of two processes:

**Theorem 7.2** (Non-terminating system analysis soundness). Given a decoupled system trace  $ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{n-1}} ic_n \wedge ic_1 \xrightarrow{\beta_1} ic_2 \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} ic_n$  from some state  $ic_1 = \langle s_1, \rho_1, h_1 \rangle$  to some state  $ic_n = \langle s_n, \rho_n, h_n \rangle$  and assuming  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1, \rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_\rho(first(s_1))), |h_1| \in \mathcal{L}(\widehat{\mathcal{E}}_h(first(s_1))), and |\beta_1\beta_2\dots\beta_{n-1}f| \in \mathcal{L}(\widehat{\mathcal{E}}_f(first(s_1)))$  then

$$\rho_n \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_n))),$$
  
$$|h_1\alpha_1\alpha_2\dots\alpha_{n-1}| = |h_n| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_n))), \text{ and}$$
  
$$|f| \in \gamma_{st}(\widehat{\mathcal{E}}_{f}(first(s_n)))$$

From the general theorem we can then extract the following corollary which states soundness of any reachable state and thereby proves why the collective history  $\bigsqcup_{\ell} \widehat{\mathcal{E}_h}(\ell)$  is sufficient in our analysis algorithms. By symmetry the corresponding property allows us to analyze and extract a sound collective history from  $s_2$ .

**Corollary 7.1** (Analysis soundness). Given a decoupled system trace  $ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{n-1}} ic_n \wedge ic_1 \xrightarrow{\beta_1} ic_2 \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} ic_n$  from some initial state  $ic_1 = \langle s_1, \rho_1, \epsilon \rangle$  to any reachable state  $ic_n = \langle s_n, \rho_n, h_n \rangle$  and assuming  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and sound approximations of

- the initial environment  $\rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1))),$
- the initial history  $\epsilon \in \mathcal{L}(\widehat{\mathcal{E}_h}(first(s_1)))$ , and of
- the surrounding communication

$$\beta_1\beta_2\dots\beta_{n-1} \in \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_1)))$$

then the entry point  $\widehat{\mathcal{E}}(first(s_n))$  soundly approximates

- the reachable stores  $\rho_n \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_n))))$ ,
- the communication history

$$|\alpha_1 \alpha_2 \dots \alpha_{n-1}| = |h_n| \in \mathcal{L}(\mathcal{E}_h(first(s_n)))$$
 and

• the communication future  $\epsilon \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_n)))).$ 

The alert reader may have noticed that this expresses soundness of any reachable, non-terminal configuration. As such, for a terminating process, e.g.,  $\langle ch!^{\ell}e, \rho, \epsilon \rangle \xrightarrow{ch!v} \langle \rho, ch!v \rangle$  there is no non-terminal configuration  $\langle s^{\ell'}, \rho, h \rangle$  from which we can pick up the last network activity ch!v in  $\widehat{\mathcal{E}}_{h}(\ell')$  as a precondition. We circumvent this issue by inserting a dummy skip statement at the end of each process. As a consequence, a process such as the above now rewrites to a non-terminal configuration (with the dummy statement):  $\langle ch!^{\ell}e; \mathrm{skip}^{\ell'}, \rho, \epsilon \rangle \xrightarrow{ch!v} \langle \mathrm{skip}^{\ell'}, \rho, ch!v \rangle$  from which we can pick up the last network activity ch!v in  $\widehat{\mathcal{E}}_{h}(\ell')$  as the dummy statement's precondition.

# 8. Related Work

Static analysis of processes already has a rich history. Cousot and Cousot (1980) initially extended the abstract interpretation framework to analyze communicating sequential processes. Our work differs in that the domain of LREs can express temporal properties ('a value is read from channel ch2 after a value is written to channel ch1'), whereas the domains of Cousot and Cousot (1980) do not. Since then a range of papers have investigated static analyses of process algebras and mobile calculi. For example, Venet (1998) developed a static analysis framework for  $\pi$ -calculus, Rydhof Hansen et al. (1999) develop a control-flow analysis and an occurrence counting analysis for mobile ambients, and Feret developed static analyses (control-flow analysis and occurrence counting analysis) for  $\pi$ -calculus (Feret 2000) and later for mobile ambients (Feret 2001) in the form of a parameterized control-flow analysis. Instead of analyzing such high-level process models, we focus on extending standard analyses to also analyze the order and the *content* of network communication such that the knowledge from traditional domains and LREs may benefit from each other.

Recently, Miné has developed a static analysis for programs with shared-memory concurrency (Miné 2014). Both Miné's approach and our approach share the idea of extending (or lifting) existing analysis techniques to deal with concurrency. In contrast to the present analysis that focuses on inferring message content (and order) Miné's analysis gradually infers the *interference* of individual threads, i.e., the values stored into shared locations. His analysis is thread modular, meaning that it can analyze a thread in isolation based on the interference of other threads without having to consider their source code. This aspect is reminiscent of how we analyze one process in isolation based on its future. Miné's work furthermore builds on viewing rely-guaranteee reasoning as an abstract interpretation, a view which is then used to infer *rely*assertions. Similarly one can view our process futures as a form of inferred rely-assertions.

The present paper builds on Midtgaard et al. (2016), in which we developed the domain of LREs. As a use-case the previous paper devised an analysis with futures—but it lacked histories. As a consequence the analysis was able only to analyze a single process against a manually written network policy given by an LRE (a future). Furthermore the work lacked a formal soundness proof. The current paper improves on the previous paper by devising an analysis that removes the need for any manual policy specification and furthermore proves that iterative algorithms that repeat it are sound with respect to an operational semantics.

Abstract interpretation has a history of iterative analysis algorithms over decreasing chains. Examples include Bourdoncle (1993)'s iterated forward-backward analysis for statically debugging, Cousot and Cousot (1992a)'s iterated forward-backward analysis algorithm, and Logozzo (2004)'s analysis algorithm for analyzing object-oriented classes against an environment of client callers. Our analysis approach is inspired by the latter. Logozzo (2004) devises a modular analysis of class invariants using contexts approximated by a lattice-valued regular expression domain to capture calling policies. In Logozzo's work the ability to analyze classes separately is however central, with concurrency being less of a concern. Under the view of objects as processes that send and receive messages, the two approaches are however clearly related.

Our analysis depends on LREs: a regular language domain. A similar domain of *lattice automata* was developed by Le Gall et al. (2006) and Le Gall and Jeannet (2007). Both LREs and lattice automata depend on atomistic lattices. The two parametric domains furthermore echo the well-known correspondence between "plain" regular languages and finite automata. Given several decades of work on automata and process algebra to capture in an abstract model phenomena of software processes that lends itself to formal reasoning, a natural next step is to push for automated reasoning, e.g., static analysis, over suitable lattice-valued generalizations of such structures.

In terms of expressivity, the kinds of properties we can infer with LREs could also be expressed with *session types* (Dezani-Ciancaglini and de'Liguoro 2009). However in order to also express the content of messages session types would have to be suitably extended with a form of *refinement types* (Freeman and Pfenning 1991) or *liquid types* (Rondon et al. 2008) to express message contents. We stress that the current approach *infers* the network communication automatically rather than just check it. Inspired by how *multiparty session types* (Honda et al. 2008) lifted session types's previous restriction to two-party sessions, a natural next step would be to extend the current analysis approach beyond two parties as well.

# 9. Conclusion and Future Work

We have developed a static analysis targeting safety properties of concurrent, message-passing programs. The approach combines traditional abstract stores and lattice-valued regular expressions that capture both order and content of network communication. In the longer run we envision that the approach can be extended to perform an iterated forwards/backwards analysis for static debugging of message-passing programs along the lines of Bourdoncle. Another avenue we would like to pursue is an extension of the approach to analyze the order and message content of a language with asynchronous communication.

# Acknowledgments

The authors thank the anonymous reviewers for constructive comments and suggestions. This work was supported by IDEA4CPS, granted by the Danish Research Foundations for Basic Research (DNRF86-10).

# References

- A. Aiken. Introduction to set constraint-based program analysis. Science of Computer Programming, 35(2-3):79–111, 1999.
- F. Bourdoncle. Abstract debugging of higher-order imperative languages. In D. W. Wall, editor, *Proc. of the ACM SIGPLAN 1993 Conference* on Programming Languages Design and Implementation, pages 46–55, June 1993.
- J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.
- P. Cousot. Semantic foundations of program analysis. In S. S. Muchnick and N. D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, 1981.
- P. Cousot. Abstracting induction by extrapolation and interpolation. In D. D'Souza, A. Lal, and K. G. Larsen, editors, Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Proceedings, volume 8931 of LNCS, pages 19–42, 2015.

- P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In Proc. of the Second International Symposium on Programming, pages 106–130, 1976.
- P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In R. Sethi, editor, *Proc. of the Fourth Annual ACM Symposium* on Principles of Programming Languages, pages 238–252, Jan. 1977.
- P. Cousot and R. Cousot. Semantic analysis of Communicating Sequential Processes. In J. W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout*, volume 85 of *LNCS*, pages 119–133, The Netherland, July 1980.
- P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2–3):103–179, 1992a.
- P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proc. of the 4th International Symposium* on Programming Language Implementation and Logic Programming (*PLILP '92*), volume 631 of *LNCS*, pages 269–295, Leuven, Belgium, Aug. 1992b. Springer-Verlag.
- B. A. Davey and H. A. Priestley. Introduction to Lattices and Order. Cambridge University Press, Cambridge, England, second edition, 2002.
- M. Dezani-Ciancaglini and U. de'Liguoro. Sessions and session types: An overview. In C. Laneve and J. Su, editors, Web Services and Formal Methods, 6th International Workshop, WS-FM 2009, Revised Selected Papers, volume 6194 of LNCS, pages 1–28, 2009.
- J. Feret. Confidentiality analysis of mobile systems. In J. Palsberg, editor, Static Analysis, 7th International Symposium, SAS 2000, volume 1824 of LNCS, pages 135–154, Jun-Jul 2000.
- J. Feret. Abstract interpretation-based static analysis of mobile ambients. In P. Cousot, editor, *Static Analysis, 8th International Symposium, SAS 2001*, volume 2126 of *LNCS*, pages 412–430, Jul 2001.
- T. Freeman and F. Pfenning. Refinement types for ML. In B. Ryder, editor, Proc. of the ACM SIGPLAN 1991 Conference on Programming Languages Design and Implementation, pages 268–277, June 1991.
- G. Grätzer. *General Lattice Theory*. Pure and Applied Mathematics. Academic Press, 1978.
- N. Halbwachs. Delay analysis in synchronous programs. In C. Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV '93, Proceedings*, volume 697 of *LNCS*, pages 333–346, 1993.
- K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In G. C. Necula and P. Wadler, editors, *Proc. of the 35th Annual* ACM Symposium on Principles of Programming Languages, pages 273– 284, Jan. 2008.
- T. Le Gall and B. Jeannet. Lattice automata: A representation for languages on infinite alphabets, and some applications to verification. In H. R. Nielson and G. Filé, editors, *Static Analysis, 14th International Symposium, SAS 2007*, volume 4634 of *LNCS*, pages 52–68, Aug. 2007.
- T. Le Gall, B. Jeannet, and T. Jéron. Verification of communication protocols using abstract interpretation of FIFO queues. In M. Johnson and V. Vene, editors, *Proc. of the 11th International Conference on Algebraic Methodology and Software Technology, AMAST '06*, volume 4019 of *LNCS*, pages 204–219, 2006.
- F. Logozzo. Separate compositional analysis of class-based object-oriented languages. In C. Rattray, S. Maharaj, and C. Shankland, editors, *Proc.* of the 10th International Conference on Algebraic Methodology and Software Technology, AMAST '04, volume 3116 of LNCS, pages 334– 348, 2004.
- J. Midtgaard, F. Nielson, and H. R. Nielson. A parametric abstract domain for lattice-valued regular expressions. In X. Rival, editor, *Static Analysis, 23rd International Symposium, SAS 2016*, volume 9837 of *LNCS*, Edinburgh, UK, Sept. 2016. Springer-Verlag.
- A. Miné. Relational thread-modular static value analysis by abstract interpretation. In K. L. McMillan and X. Rival, editors, Verification, Model Checking, and Abstract Interpretation, 15th International Conference, VMCAI 2014, Proc., volume 8318 of LNCS, pages 39–58. Springer-Verlag, Jan. 2014.

- D. Monniaux. A minimalistic look at widening operators. Higher-Order and Symbolic Computation, 22(2):145-154, 2009.
- F. Nielson, H. R. Nielson, and C. Hankin. Principles of Program Analysis. Springer, 1999.
- H. R. Nielson and F. Nielson. Flow logic: a multi-paradigmatic approach to static analysis. In T. Æ. Mogensen, D. A. Schmidt, and I. H. Sudborough, editors, The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones, volume 2566 of LNCS, pages 223-244, 2002.
- P. M. Rondon, M. Kawaguci, and R. Jhala. Liquid types. In S. Amarasinghe, editor, Proc. of the ACM SIGPLAN 2008 Conference on Programming Languages Design and Implementation, pages 159-169, June 2008.
- R. Rydhof Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In A. Cortesi and G. Filé, editors, Static Analysis, 6th International Symposium, SAS '99, volume 1694 of LNCS, pages 134-148, Sept. 1999.
- D. Sangiorgi. On the origins of bisimulation and coinduction. ACM Transactions on Programming Languages and Systems, 31(4), 2009.
- V. T. Vasconcelos, S. Gay, and A. Ravara. Typechecking a multithreaded functional language with session types. Theoretical Computer Science, 368(1-2):64-87, 2006.
- A. Venet. Automatic determination of communication topologies in mobile systems. In G. Levi, editor, Static Analysis, 5th International Symposium, SAS '98, volume 1503 of LNCS, pages 152-167, Sept. 1998.

#### Wellformedness proofs A.

#### Read and write abstractions are atom-preserving Galois A.1 insertions

*Proof.* Let  $S \subseteq Channel \times \{?\} \times Val$  and  $([l; u], \hat{v}) \in Interval *$ *Val* be given.

$$\begin{array}{c} (c) & (c)$$

$$\Rightarrow S \subseteq \gamma_{rd}([l; u], \hat{v})$$
 (by def. of  $\gamma_{rd}$ )

Since  $\alpha_{Int}$  and  $\alpha_v$  are onto, so is  $\alpha_{rd}$  by definition. Finally since  $\alpha_{Int}: Atoms(\mathbb{Z}) \longrightarrow Atoms(Interval) \text{ and } \alpha_v: Atoms(\mathbb{Z}) \longrightarrow$ Atoms(Val) by assumption, we have

 $\alpha_{rd}: Atoms(\wp(Channel \times \{?\} \times Val)) \longrightarrow Atoms(Interval * \widehat{Val})$ The proof for write is identical (up to renaming). 

#### A.2 Channel abstraction is an atom-preserving Galois insertion

*Proof.* Let  $S \subseteq Action$ ,  $(\hat{v}_r, \hat{v}_w) \in \widehat{Ch}(\widehat{Val})$  be given.

$$\begin{aligned} \alpha_{ch}(S) &\sqsubseteq (\hat{v}_{r}, \hat{v}_{w}) \\ \iff \alpha_{rd}(\{ch?v \in S\}) &\sqsubseteq \hat{v}_{r} \land \alpha_{wr}(\{ch!v \in S\}) \sqsubseteq \hat{v}_{w} \\ & (by \text{ def. of } \alpha_{ch}) \end{aligned}$$
$$\\ \iff \{ch?v \in S\} \subseteq \alpha_{rd}(\hat{v}_{r}) \land \{ch!v \in S\} \subseteq \gamma_{wr}(\hat{v}_{w}) \\ & (Galois \text{ conn.}) \end{aligned}$$
$$\\ \iff S \subseteq \gamma_{rd}(\hat{v}_{r}) \cup \gamma_{wr}(\hat{v}_{w}) \\ \iff S \subseteq \gamma_{ch}(\hat{v}_{r}, \hat{v}_{w}) \end{aligned}$$
$$(by \text{ def. of } Action)$$

If  $\alpha_{wr}$  and  $\alpha_{rd}$  are onto, so is  $\alpha_{ch}$  by definition. Finally since

 $\alpha_{rd}: Atoms(\wp(Channel \times \{?\} \times Val)) \longrightarrow Atoms(Interval * \widehat{Val})$ 

 $\alpha_{wr}: Atoms(\wp(Channel \times \{!\} \times Val)) \longrightarrow Atoms(Interval * \widehat{Val})$ 

we have  $\alpha_{rd}$ :  $Atoms(\wp(Action)) \longrightarrow Atoms(\widehat{Ch}(\widehat{Val}))$  since a singleton set must be either a read or a write. 

# A.3 $\widehat{\mathcal{A}}$ monotone

*Proof.* By structural induction on *e*. Let *e* and  $\hat{\rho} \stackrel{.}{\sqsubseteq} \hat{\rho}'$  be given.

**case** *n*: By definition of  $\widehat{\mathcal{A}}$ :  $\widehat{\mathcal{A}}(n, \widehat{\rho}) = \alpha_v(\{n\}) = \widehat{\mathcal{A}}(n, \widehat{\rho}')$  **case** *x*: If  $\widehat{\rho} = \bot$  then by definition of  $\widehat{\mathcal{A}}$ :  $\widehat{\mathcal{A}}(x, \widehat{\rho}) = \bot \sqsubseteq \widehat{\mathcal{A}}(x, \widehat{\rho}')$ If  $\widehat{\rho} \neq \bot$  then by definition of  $\widehat{\mathcal{A}}$  and assumption:  $\widehat{\mathcal{A}}(x, \widehat{\rho}) = \widehat{\rho}(x) \sqsubseteq \widehat{\rho}'(x) = \widehat{\mathcal{A}}(x, \widehat{\rho}')$  **case** ?: By definition of  $\widehat{\mathcal{A}}$ :  $\widehat{\mathcal{A}}(?, \widehat{\rho}) = \top = \widehat{\mathcal{A}}(?, \widehat{\rho}')$  **case**  $e_1 + e_2$ :  $\widehat{\mathcal{A}}(e_1 + e_2, \widehat{\rho}) = \widehat{\mathcal{A}}(e_1, \widehat{\rho}) + \widehat{\mathcal{A}}(e_2, \widehat{\rho})$  (by def. of  $\widehat{\mathcal{A}}$ )  $\sqsubseteq \widehat{\mathcal{A}}(e_1, \widehat{\rho}') + \widehat{\mathcal{A}}(e_2, \widehat{\rho})$  (by the IH, monotonicity of  $\widehat{+}$ )  $\sqsubseteq \widehat{\mathcal{A}}(e_1, \widehat{\rho}') + \widehat{\mathcal{A}}(e_2, \widehat{\rho}')$  (by the IH, monotonicity of  $\widehat{+}$ )  $= \widehat{\mathcal{A}}(e_1 + e_2, \widehat{\rho}')$  (by def. of  $\widehat{\mathcal{A}}$ ) **case**  $e_1 - e_2$ :

$$\widehat{\mathcal{A}}(e_1 - e_2, \widehat{\rho}) = \widehat{\mathcal{A}}(e_1, \widehat{\rho}) - \widehat{\mathcal{A}}(e_2, \widehat{\rho}) \qquad \text{(by def. of } \widehat{\mathcal{A}})$$

$$\sqsubseteq \widehat{\mathcal{A}}(e_1, \widehat{\rho}') - \widehat{\mathcal{A}}(e_2, \widehat{\rho}) \qquad \text{(by the IH, monotonicity of } \widehat{-})$$

$$\sqsubseteq \widehat{\mathcal{A}}(e_1, \widehat{\rho}') - \widehat{\mathcal{A}}(e_2, \widehat{\rho}') \qquad \text{(by the IH, monotonicity of } \widehat{-})$$

$$= \widehat{\mathcal{A}}(e_1 - e_2, \widehat{\rho}') \qquad \text{(by def. of } \widehat{\mathcal{A}})$$

# In first argument

*Proof.* Let  $\hat{\rho} \stackrel{.}{\sqsubseteq} \hat{\rho}'$  and  $x, \hat{v}$  be given. Now:  $\widehat{assign}(\hat{\rho}, x, \hat{v}) = \hat{\rho}[x \mapsto \hat{v}] \stackrel{.}{\sqsubseteq} \hat{\rho}'[x \mapsto \hat{v}] = \widehat{assign}(\hat{\rho}', x, \hat{v})$ 

## In third argument

*Proof.* Let  $\hat{\rho}, x$ , and  $\hat{v} \subseteq \hat{v}'$  be given. Now:  $\widehat{assign}(\hat{\rho}, x, \hat{v}) = \hat{\rho}[x \mapsto \hat{v}] \subseteq \hat{\rho}[x \mapsto \hat{v}'] = \widehat{assign}(\hat{\rho}, x, \hat{v}')$ 

### A.5 *true* monotone

Note: the following proof concerns true for the interval lattice. For the parity lattice the proof is identical except for the last subcase which is simpler, due to the simpler definition.

*Proof.* By structural induction on b. Let b and  $\hat{\rho} \sqsubseteq \hat{\rho}'$  be given.

**case tt:** By definition of  $\widehat{true}$ :  $\widehat{true}(\texttt{tt}, \widehat{\rho}) = \widehat{\rho} \stackrel{:}{\sqsubseteq} \widehat{\rho'} = \widehat{true}(\texttt{tt}, \widehat{\rho'})$ **case ff:** By definition of  $\widehat{true}$ :  $\widehat{true}(\texttt{ff}, \widehat{\rho}) = \bot = \widehat{true}(\texttt{ff}, \widehat{\rho'})$ **case**  $x_1 < x_2$ : We consider each case of  $\widehat{true}$ :

subcase  $\hat{\rho} = \bot \lor \hat{\rho}(x_1) = \bot \lor \hat{\rho}(x_2) = \bot$ : By definition of  $\widehat{true}$ :

 $\widehat{true}(x_1 < x_2, \widehat{\rho}) = \bot \stackrel{.}{\sqsubseteq} \widehat{true}(x_1 < x_2, \widehat{\rho}')$ 

subcase  $\widehat{\rho}(x_1) = [l_1; u_1] \land \widehat{\rho}(x_2) = [l_2; u_2]$ : By assumption we have  $\widehat{\rho}(x_1) = [l_1; u_1] \sqsubseteq \widehat{\rho}'(x_1) = [l'_1; u'_1]$  and Note: the following proof concerns false for the interval lattice. For the parity lattice the proof is identical except for the last subcase which is simpler, due to the simpler definition.

# A.6 *false* monotone

*Proof.* By structural induction on b. Let  $\hat{\rho} \stackrel{.}{\sqsubseteq} \hat{\rho}'$  be given.

**case** tt: By definition of  $\widehat{false}$ :  $\widehat{false}(tt, \hat{\rho}) = \bot = \widehat{false}(tt, \hat{\rho}')$ **case ff:** By definition of  $\widehat{false}$ :  $\widehat{false}(ff, \widehat{\rho}) = \widehat{\rho} \stackrel{.}{\sqsubseteq} \widehat{\rho}' = \widehat{false}(ff, \widehat{\rho}')$ **case**  $x_1 < x_2$ : We consider each case of *false*: subcase  $\hat{\rho} = \bot \lor \hat{\rho}(x_1) = \bot \lor \hat{\rho}(x_2) = \bot$ : By definition of false: 
$$\begin{split} &\widehat{false}(x_1 < x_2, \widehat{\rho}) = \bot \stackrel{:}{\sqsubseteq} \widehat{false}(x_1 < x_2, \widehat{\rho}') \\ & \text{subcase } \widehat{\rho}(x_1) = [l_1; u_1] \land \widehat{\rho}(x_2) = [l_2; u_2] \text{: By} \quad \text{assumption} \\ & \text{we have } \widehat{\rho}(x_1) = [l_1; u_1] \stackrel{!}{\sqsubseteq} \stackrel{i}{\rho}'(x_1) = [l_1'; u_1'] \text{ and} \\ & \widehat{\rho}(x_2) = [l_2; u_2] \stackrel{!}{\sqsubseteq} \stackrel{i}{\rho}'(x_2) = [l_2'; u_2']. \text{ Now:} \end{split}$$
 $\widehat{false}(x_1 < x_2, \widehat{\rho})$  $= \widehat{\rho}[x_1 \mapsto [\max l_1 l_2; u_1], x_2 \mapsto [l_2; \min u_1 u_2]]$ (by def of false)  $\stackrel{.}{\sqsubseteq} \widehat{\rho}'[x_1 \mapsto [\max l_1 l_2; u_1], x_2 \mapsto [l_2; \min u_1 u_2]]$ (by assumption)  $\stackrel{.}{\sqsubseteq} \widehat{\rho}'[x_1 \mapsto [\max l_1 l_2; u_1'], x_2 \mapsto [l_2'; \min u_1 u_2]]$ (by above)  $\stackrel{.}{\sqsubseteq} \widehat{\rho}'[x_1 \mapsto [\max l_1' l_2'; u_1'], x_2 \mapsto [l_2'; \min u_1 u_2]]$ (by above)  $\stackrel{.}{\sqsubseteq} \widehat{\rho}'[x_1 \mapsto [\max l_1' l_2'; u_1'], x_2 \mapsto [l_2'; \min u_1' u_2']]$ (by above)  $= \widehat{false}(x_1 < x_2, \widehat{\rho}')$ (by def of  $\widehat{false}$ ) 

# A.7 Moore family failure

Proof.

Assume for the sake of contradiction that  $\{\mathcal{L}(r) \mid r \in \widehat{R}_A\}$  constitutes a Moore family:

# 1. It contains top:

 $\mathcal{L}(\top^*) = \bigcup_{i \ge 0} \mathcal{L}(\top)^i$  $= \bigcup_{i \ge 0} \{c \mid c \in \gamma(\top)\}^i = \bigcup_{i \ge 0} \{c \mid c \in C\}^i = C^*$ 

2. But it is not closed under arbitrary intersection. In particular, the result of the following infinite meet:

 $(even^* \cdot odd^*)$ 

 $\& (\epsilon + even \cdot even^* \cdot odd^* \cdot odd)$ 

 $\& (\epsilon + even \cdot odd + even \cdot even^* \cdot odd^* \cdot odd \cdot odd) \\ \& \dots$ 

is not in  $\widehat{R}_A$ , contradicting our Moore family assumption.

# A.8 Top is a valid analysis solution

*Proof.* By structural induction on  $s^{\ell}$ . Let  $s^{\ell}$  be given.

 $\begin{aligned} & \operatorname{case \, skip}^{\ell} \colon \operatorname{Immediately \, holds \, since \, } \widehat{\mathcal{E}^{\top}}(\ell) = (\lambda x. \top_{A}, \top^{*}, \top^{*}) = \\ & \widehat{\mathcal{X}^{\top}}(\ell). \end{aligned} \\ & \operatorname{case \, } x :=^{\ell} e \colon \operatorname{Also \, holds \, since \, } (\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(e, \widehat{\rho})), \widehat{h}, \widehat{f}) \quad \sqsubseteq \\ & (\lambda x. \top_{A}, \top^{*}, \top^{*}) = \widehat{\mathcal{X}^{\top}}(\ell) \text{ for any } (\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}^{\top}}(\ell). \end{aligned} \\ & \operatorname{case \, } s_{1}^{\ell_{1}}; s_{2}^{\ell_{2}} \colon \operatorname{By \, the \, induction \, hypothesis \, } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_{1}^{\ell_{1}} \text{ and} \\ & \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_{2}^{\ell_{2}} \text{ and since } \widehat{\mathcal{E}^{\top}}(\ell) = \widehat{\mathcal{E}^{\top}}(\ell_{1}), \widehat{\mathcal{X}^{\top}}(\ell_{1}) = \widehat{\mathcal{E}^{\top}}(\ell_{2}), \\ & \operatorname{and \, } \widehat{\mathcal{X}^{\top}}(\ell_{2}) = \widehat{\mathcal{X}^{\top}}(\ell) \text{ we have } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_{1}^{\ell_{1}}; s_{2}^{\ell_{2}}. \end{aligned} \\ & \operatorname{case \, if \, } b^{\ell} \operatorname{then \, } s_{1}^{\ell_{1}} \operatorname{else \, } s_{2}^{\ell_{2}} : \operatorname{By \, the \, induction \, hypothesis \, } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_{1}^{\ell_{1}} \operatorname{and \, } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_{2}^{\ell_{2}} \text{ and since \, } \widehat{\mathcal{X}^{\top}}(\ell_{1}) = \widehat{\mathcal{X}^{\top}}(\ell) = \widehat{\mathcal{X}^{\top}}(\ell_{2}) \\ & \operatorname{and \, } (\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \ \sqsubseteq \, \\ & \widehat{\mathcal{E}^{\top}}(\ell_{1}) \text{ and } (\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \ \sqsubseteq \\ & \widehat{\mathcal{E}^{\top}}(\ell_{2}) \text{ for \, any \, } (\widehat{\rho}, \widehat{h}, \widehat{f}) \ = \, \\ & \widehat{\mathcal{E}^{\top}}(\ell) \text{ we \, have \, } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models \end{aligned}$ 

 $\begin{array}{l} c \quad (c_2) \text{ for any } (\rho, h, f) = c \quad (c) \text{ we have } c \quad (\lambda^{-1} + if b^\ell \text{ then } s_1^{\ell_1} \text{ else } s_2^{\ell_2}. \\ \text{case while } b^\ell \text{ do } s_1^{\ell_1} \text{ end: By the induction hypothesis } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models s_1^{\ell_1} \text{ and since } \widehat{\mathcal{X}^{\top}}(\ell_1) = \widehat{\mathcal{E}^{\top}}(\ell) \text{ and } (\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{E}^{\top}}(\ell_1) \text{ and } (\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{X}^{\top}}(\ell) \text{ for any } (\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}^{\top}}(\ell) \text{ we have } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \models \text{ while } b^\ell \text{ do } s_1^{\ell_1} \text{ end.} \end{array}$ 

 $\begin{array}{l} \operatorname{case} s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2} \text{: By the induction hypothesis } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash s_1^{\ell_1} \text{ and} \\ \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash s_2^{\ell_2} \text{ and since } \widehat{\mathcal{X}^{\top}}(\ell_1) = \widehat{\mathcal{X}^{\top}}(\ell) = \widehat{\mathcal{X}^{\top}}(\ell_2) \text{ and} \\ \widehat{\mathcal{E}^{\top}}(\ell_1) = \widehat{\mathcal{E}^{\top}}(\ell) = \widehat{\mathcal{E}^{\top}}(\ell_2) \text{ we have } \widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}. \end{array}$ 

- **case**  $ch?^{\ell}x$ **:** The right-hand-side of the implication  $(\widehat{assign}(\hat{\rho}, x, \hat{v}), \hat{h} \cdot ch?\hat{v}, \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_a])}(\hat{f})) \sqsubseteq \widehat{\mathcal{X}^{\top}}(\ell)$  holds vacuously and therefore  $\widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash ch?^{\ell}x$ .
- case  $ch!^{\ell}e$ : The right-hand-side of the implication

$$(\widehat{\rho}, \widehat{h} \cdot ch! (\widehat{v} \sqcap \widehat{v}'), \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_a])}(\widehat{f})) \sqsubseteq \widehat{\mathcal{X}^{\top}}(\ell)$$

holds vacuously and therefore  $\widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash ch!^{\ell} e$ . **case** stop<sup> $\ell$ </sup>: We have  $(\perp, \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{X}^{\top}}(\ell)$  for any  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}^{\top}}(\ell)$  and therefore  $\widehat{\mathcal{E}^{\top}}, \widehat{\mathcal{X}^{\top}} \vDash \operatorname{stop}^{\ell}$ .

# A.9 Meet preserves solutions

*Proof.* By structural induction on *s*. Let  $s, \hat{\mathcal{E}}, \hat{\mathcal{X}}, \hat{\mathcal{E}}', \hat{\mathcal{X}}'$  be given. and assume  $\hat{\mathcal{E}}, \hat{\mathcal{X}} \models s$  and  $\hat{\mathcal{E}}', \hat{\mathcal{X}}' \models s$ .

- **case** skip<sup> $\ell$ </sup>: By assumption we have  $\widehat{\mathcal{E}}(\ell) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{E}}'(\ell) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$ , but then  $\widehat{\mathcal{E}}(\ell) \sqcap \widehat{\mathcal{E}}'(\ell) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{E}}(\ell) \sqcap \widehat{\mathcal{E}}'(\ell) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$ . Since  $\widehat{\mathcal{E}}(\ell) \sqcap \widehat{\mathcal{E}}'(\ell)$  is a lower bound of both  $\widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{X}}'(\ell)$ , it is less or equal to the greatest lower bound of the two:  $\widehat{\mathcal{E}}(\ell) \sqcap \widehat{\mathcal{E}}'(\ell) \sqsubseteq \widehat{\mathcal{X}}(\ell) \sqcap \widehat{\mathcal{X}}'(\ell)$ .
- **case**  $x := {}^{\ell}e$ : By assumption  $(\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(e, \widehat{\rho})), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{X}}(\ell)$ and  $(\widehat{assign}(\widehat{\rho}', x, \widehat{\mathcal{A}}(e, \widehat{\rho}')), \widehat{h}', \widehat{f}') \sqsubseteq \widehat{\mathcal{X}}'(\ell)$  where  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$  and  $(\widehat{\rho}', \widehat{h}', \widehat{f}') = \widehat{\mathcal{E}}'(\ell)$ . By monotonicity of  $\widehat{\mathcal{A}}$  we have both  $\widehat{\mathcal{A}}(e, \widehat{\rho} \sqcap \widehat{\rho}') \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho})$  and  $\widehat{\mathcal{A}}(e, \widehat{\rho} \sqcap \widehat{\rho}') \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho}')$ .

By the above and by the definition of  $\widehat{assign}$  it now follows that

$$\begin{split} & (\widehat{assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{\mathcal{A}}(e, \widehat{\rho} \sqcap \widehat{\rho}')), \widehat{h} \And \widehat{h}', \widehat{f} \And \widehat{f}') \\ & \sqsubseteq (\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(e, \widehat{\rho})), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{X}}(\ell) \\ & \text{and} \ (\widehat{assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{\mathcal{A}}(e, \widehat{\rho} \sqcap \widehat{\rho}')), \widehat{h} \And \widehat{h}', \widehat{f} \And \widehat{f}') \\ & \sqsubseteq (\widehat{assign}(\widehat{\rho}', x, \widehat{\mathcal{A}}(e, \widehat{\rho}')), \widehat{h}', \widehat{f}') \sqsubseteq \widehat{\mathcal{X}'}(\ell) \end{split}$$

As a consequence the left-hand-side is less or equal to the greatest lower bound of the two right-hand-sides  $\widehat{\mathcal{X}}(\ell) \sqcap \widehat{\mathcal{X}}'(\ell)$ . The meet thereby satisfies the analysis specification as it is defined componentwise:  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) = \widehat{\mathcal{E}}(\ell) \sqcap \widehat{\mathcal{E}}'(\ell) = (\widehat{\rho} \sqcap \widehat{\rho}', \widehat{h} \& \widehat{h}', \widehat{f} \& \widehat{f}')$ .

- **case**  $s_1$ ;  $s_2$ : By assumption we have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1, \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_2, \widehat{\mathcal{E}}', \widehat{\mathcal{X}'} \models s_1, \widehat{\mathcal{E}}', \widehat{\mathcal{X}'} \models s_2, \text{ and } \widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(first(s_2)) \text{ and } \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}'(first(s_2)) \text{ for all } \ell_1 \in last(s_1).$  From the first part it follows from the induction hypothesis that  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}'), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_1, \text{ and } (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}'), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_2.$  From the second part it follows that  $\widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(first(s_2))$  and  $\widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}'(first(s_2))$  for any  $\ell_1 \in last(s_1)$ . Hence  $(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_1) = \widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(first(s_2)) \sqcap \widehat{\mathcal{E}}'(first(s_2)) = (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(first(s_2))$
- **case** if  $b^{\ell}$  then  $s_1$  else  $s_2$ : By assumption we have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$ ,  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_2, \widehat{\mathcal{E}'}, \widehat{\mathcal{X}'} \models s_1$ , and  $\widehat{\mathcal{E}'}, \widehat{\mathcal{X}'} \models s_2$ . By two applications of the induction hypothesis we therefore get  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}'}), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_1$  and  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}'}), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_2$ .

By assumption we also have  $(\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{E}}(first(s_1)),$  $(\widehat{false}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{E}}(first(s_2)), (\widehat{true}(b, \widehat{\rho}'), \widehat{h}', \widehat{f}') \sqsubseteq \widehat{\mathcal{E}}'(first(s_1)),$  and  $(\widehat{false}(b, \widehat{\rho}'), \widehat{h}', \widehat{f}') \sqsubseteq \widehat{\mathcal{E}}'(first(s_2))$  where  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$  and  $(\widehat{\rho}', \widehat{h}', \widehat{f}') = \widehat{\mathcal{E}}'(\ell)$ . By definition of  $\sqcap$ we have  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) = (\widehat{\rho}' \sqcap \widehat{\rho}', \widehat{h} \& \widehat{h}', \widehat{f} \& \widehat{f}')$ . Hence

$$\begin{split} &(\widehat{true}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}\&\widehat{h}',\widehat{f}\&\widehat{f})\sqsubseteq\widehat{\mathcal{E}}(first(s_1))\\ &(\widehat{true}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}&\widehat{h}',\widehat{f}&\widehat{f})\sqsubseteq\widehat{\mathcal{E}}'(first(s_1))\\ &(\widehat{false}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}&\widehat{h}',\widehat{f}&\widehat{f}')\sqsubseteq\widehat{\mathcal{E}}(first(s_2))\\ &(\widehat{false}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}&\widehat{h}',\widehat{f}&\widehat{f}')\sqsubseteq\widehat{\mathcal{E}}'(first(s_2)) \end{split}$$

by monotonicity of  $\widehat{frue}$  and  $\widehat{false}$  and by the definition  $\Box$ . Since the left-hand-sides are lower bounds they are also less

or equal to the greatest lower bounds:

$$(\widehat{true}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}\&\widehat{h}',\widehat{f}\&\widehat{f}) \sqsubseteq \widehat{\mathcal{E}}(first(s_1)) \sqcap \widehat{\mathcal{E}}'(first(s_1)) = (\widehat{\mathcal{E}}\sqcap\widehat{\mathcal{E}}')(first(s_1)) (\widehat{false}(b,\widehat{\rho}\sqcap\widehat{\rho}'),\widehat{h}\&\widehat{h}',\widehat{f}\&\widehat{f}') \sqsubseteq \widehat{\mathcal{E}}(first(s_2)) \sqcap \widehat{\mathcal{E}}'(first(s_2)) = (\widehat{\mathcal{E}}\sqcap\widehat{\mathcal{E}}')(first(s_2))$$

Finally by assumption we have  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{X}}'(\ell_1) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$  for all  $\ell_1 \in last(s_1)$ , and  $\widehat{\mathcal{X}}(\ell_2) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{X}}'(\ell_2) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$  for all  $\ell_2 \in last(s_2)$ . But that means

$$\begin{aligned} \widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell) \text{ and } \widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{X}'}(\ell) \\ \widehat{\mathcal{X}}(\ell_2) \sqcap \widehat{\mathcal{X}'}(\ell_2) \sqsubseteq \widehat{\mathcal{X}}(\ell) \text{ and } \widehat{\mathcal{X}}(\ell_2) \sqcap \widehat{\mathcal{X}'}(\ell_2) \sqsubseteq \widehat{\mathcal{X}'}(\ell) \end{aligned}$$

for any  $\ell_1 \in last(s_1)$  and for any  $\ell_2 \in last(s_2)$ . Hence the left-hand-sides are less or equal to the greatest lower bounds:

$$\begin{aligned} &(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_1) = \widehat{\mathcal{X}}(\ell_1) \sqcap \widehat{\mathcal{X}'}(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell) \sqcap \widehat{\mathcal{X}'}(\ell) = (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell) \\ &(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_2) = \widehat{\mathcal{X}}(\ell_2) \sqcap \widehat{\mathcal{X}'}(\ell_2) \sqsubseteq \widehat{\mathcal{X}}(\ell) \sqcap \widehat{\mathcal{X}'}(\ell) = (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell) \end{aligned}$$

for any  $\ell_1 \in last(s_1)$  and for any  $\ell_2 \in last(s_2)$ .

**case** while  $b^{\ell}$  do  $s_1$  end: By assumption we have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}', \widehat{\mathcal{X}'} \models s_1$ , hence by the induction hypothesis we immediately get  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}'}), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_1$ . By assumption we furthermore have

$$\begin{split} &(\widehat{true}(b,\widehat{\rho}),\widehat{h},\widehat{f}) \sqsubseteq \widehat{\mathcal{E}}(first(s_1)) \\ &(\widehat{true}(b,\widehat{\rho}'),\widehat{h}',\widehat{f}') \sqsubseteq \widehat{\mathcal{E}}'(first(s_1)) \\ &(\widehat{false}(b,\widehat{\rho}),\widehat{h},\widehat{f}) \sqsubseteq \widehat{\mathcal{X}}(\ell) \\ &\text{and} \ (\widehat{false}(b,\widehat{\rho}'),\widehat{h}',\widehat{f}') \sqsubseteq \widehat{\mathcal{X}'}(\ell) \end{split}$$

where  $(\hat{\rho}, \hat{h}, \hat{f}) = \hat{\mathcal{E}}(\ell)$  and  $(\hat{\rho}', \hat{h}', \hat{f}') = \hat{\mathcal{E}}'(\ell)$ . But then  $(\widehat{true}(b, \hat{\rho} \sqcap \hat{\rho}'), \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq (\widehat{true}(b, \hat{\rho}), \hat{h}, \hat{f}) \sqsubseteq \hat{\mathcal{E}}(first(s_1))$   $(\widehat{true}(b, \hat{\rho} \sqcap \hat{\rho}'), \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq (\widehat{true}(b, \hat{\rho}'), \hat{h}', \hat{f}') \sqsubseteq \hat{\mathcal{E}}'(first(s_1))$   $(\widehat{false}(b, \hat{\rho} \sqcap \hat{\rho}'), \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq (\widehat{false}(b, \hat{\rho}), \hat{h}, \hat{f}) \sqsubseteq \hat{\mathcal{X}}(\ell)$  $(\widehat{false}(b, \hat{\rho} \sqcap \hat{\rho}'), \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq (\widehat{false}(b, \hat{\rho}'), \hat{h}', \hat{f}') \sqsubseteq \hat{\mathcal{X}}'(\ell)$ 

by monotonicity of  $\widehat{true}$  and  $\widehat{false}$ . Hence the left-hand-sides are less or equal to the greatest lower bound of the right-hand-sides:

$$(\widehat{true}(b,\widehat{\rho} \sqcap \widehat{\rho}'),\widehat{h} \& \widehat{h}',\widehat{f} \& \widehat{f}') \sqsubseteq (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(first(s_1))$$
$$(\widehat{false}(b,\widehat{\rho} \sqcap \widehat{\rho}'),\widehat{h} \& \widehat{h}',\widehat{f} \& \widehat{f}') \sqsubseteq (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}}')(\ell)$$

where by definition  $(\widehat{\mathcal{E}}' \sqcap \widehat{\mathcal{E}})(\ell) = (\widehat{\rho} \sqcap \widehat{\rho}', \widehat{h} \& \widehat{h}', \widehat{f} \& \widehat{f}')$ . Finally by assumption  $\forall \ell_1 \in last(s_1)$ .  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(\ell)$  and  $\forall \ell_1 \in last(s_1)$ .  $\widehat{\mathcal{X}}'(\ell_1) \sqsubseteq \widehat{\mathcal{E}}'(\ell)$ . But then  $(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}}')(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(\ell)$  and  $(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}}')(\ell_1) \sqsubseteq \widehat{\mathcal{E}}'(\ell)$  and the left-hand-side is therefore less or equal to the greatest lower bound of the right-hand-sides:  $(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}}')(\ell_1) \sqsubseteq (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell)$  for any  $\ell_1 \in last(s_1)$ .

**case**  $s_1 \oplus^{\ell} s_2$ : By assumption we have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1, \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_2, \widehat{\mathcal{E}}', \widehat{\mathcal{X}'} \models s_1, \text{ and } \widehat{\mathcal{E}'}, \widehat{\mathcal{X}'} \models s_2.$  By two applications of the induction hypothesis we therefore get  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}'}), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_1$  and  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}'}), (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'}) \models s_2$ . By assumption we furthermore have  $\widehat{\mathcal{E}}(\ell) \sqsubseteq \widehat{\mathcal{E}}(first(s_1)), \widehat{\mathcal{E}'}(\ell) \sqsubseteq \widehat{\mathcal{E}'}(first(s_1)), \widehat{\mathcal{E}}(\ell) \sqsubseteq$ 

 $\widehat{\mathcal{E}}(\operatorname{first}(s_2))$ , and  $\widehat{\mathcal{E}'}(\ell) \sqsubseteq \widehat{\mathcal{E}'}(\operatorname{first}(s_2))$ . But then

 $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) \sqsubseteq \widehat{\mathcal{E}}(first(s_1)) \text{ and } (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) \sqsubseteq \widehat{\mathcal{E}}'(first(s_1))$  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) \sqsubseteq \widehat{\mathcal{E}}(first(s_2)) \text{ and } (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) \sqsubseteq \widehat{\mathcal{E}}'(first(s_2))$ 

and hence

$$(\widehat{\mathcal{E}} \sqcap \mathcal{E}')(\ell) \sqsubseteq (\widehat{\mathcal{E}} \sqcap \mathcal{E}')(first(s_1))$$
$$(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) \sqsubseteq (\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(first(s_2))$$

Finally by assumption  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{X}}'(\ell_1) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$  for all  $\ell_1 \in last(s_1)$  and  $\widehat{\mathcal{X}}(\ell_2) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  and  $\widehat{\mathcal{X}}'(\ell_2) \sqsubseteq \widehat{\mathcal{X}}'(\ell)$  for all  $\ell_2 \in last(s_2)$ . But that means

$$\begin{aligned} & (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell) \text{ and } (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_1) \sqsubseteq \widehat{\mathcal{X}'}(\ell) \\ & (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_2) \sqsubseteq \widehat{\mathcal{X}}(\ell) \text{ and } (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_2) \sqsubseteq \widehat{\mathcal{X}'}(\ell) \end{aligned}$$

and as a consequence

$$(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_1) \sqsubseteq (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell)$$
$$(\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell_2) \sqsubseteq (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell)$$

for all  $\ell_1 \in last(s_1)$  and for all  $\ell_2 \in last(s_2)$ . case  $ch?^{\ell}x$ : By assumption

- if  $ch!\hat{v} = \widehat{project}([ch!\hat{v}_a])$  and  $\widehat{\mathcal{D}_{repr}([ch!\hat{v}_a])}(\widehat{f}) \not\subseteq \emptyset$  then  $(\widehat{assign}(\widehat{\rho}, x, \widehat{v}), \widehat{h} \cdot ch?\hat{v}, \widehat{\mathcal{D}_{repr}([ch!\hat{v}_a])}(\widehat{f})) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  for all  $[ch!\hat{v}_a] \in \widehat{range}(\widehat{f})$  where  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$  and
- if  $ch!\hat{v}' = \widetilde{project}([ch!\hat{v}'_a])$  and  $\widehat{\mathcal{D}}_{\widetilde{repr}([ch!\hat{v}'_a])}(\widehat{f}') \not\subseteq \emptyset$ then  $(\widetilde{assign}(\widehat{\rho}', x, \widehat{v}'), \widehat{h}' \cdot ch?\hat{v}', \widehat{\mathcal{D}}_{\widetilde{repr}([ch!\hat{v}'_a])}(\widehat{f}')) \subseteq \widehat{\mathcal{X}'}(\ell)$  for all  $[ch!\hat{v}'_a] \in \widetilde{range}(\widehat{f}')$  where  $(\widehat{\rho}', \widehat{h}', \widehat{f}') = \widehat{\mathcal{E}'}(\ell)$ .

Now let  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) = (\widehat{\rho} \sqcap \widehat{\rho}', \widehat{h} \& \widehat{h}', \widehat{f} \& \widehat{f}')$  and let  $[ch!\widehat{v}_a''] \in \widehat{range}(\widehat{f} \& \widehat{f}') = \widehat{overlay}(\widehat{range}(\widehat{f}), \widehat{range}(\widehat{f}'))$  be given, and assume that  $ch!\widehat{v}'' = \widehat{project}([ch!\widehat{v}_a''])$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a''])}(\widehat{f} \& \widehat{f}') = \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a''])}(\widehat{f}) \& \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a''])}(\widehat{f}')$   $\not\in \emptyset$ . Since

$$\widehat{overlay}(\widehat{range}(\widehat{f}), \widehat{range}(\widehat{f}')) \sqsubseteq \widehat{range}(\widehat{f})$$
  
and 
$$\widehat{overlay}(\widehat{range}(\widehat{f}), \widehat{range}(\widehat{f}')) \sqsubseteq \widehat{range}(\widehat{f}')$$

under the refinement ordering there exists  $[ch!\hat{v}_a] \in \widehat{range}(\hat{f})$ and  $[ch!\hat{v}'_a] \in \widehat{range}(\hat{f}')$  such that  $[ch!\hat{v}''_a] \sqsubseteq [ch!\hat{v}_a]$  and  $[ch!\hat{v}''_a] \sqsubseteq [ch!\hat{v}'_a]$  under a sets-of-atoms ordering. Hence by monotonicity of  $\widehat{project}$ ,

$$ch!\hat{v}'' \sqsubseteq ch!\hat{v} = \widehat{project}([ch!\hat{v}_a])$$
  
and  $ch!\hat{v}'' \sqsubseteq ch!\hat{v}' = \widehat{project}([ch!\hat{v}_a'])$ 

Since  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}'_{a}])}(\widehat{f}) \& \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}''_{a}])}(\widehat{f}') \not\subseteq \emptyset$  we have both  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}''_{a}])}(\widehat{f}) \not\subseteq \emptyset$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}''_{a}])}(\widehat{f}') \not\subseteq \emptyset$ . Furthermore, by the definition of the partition being computed we have  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_{a}])}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}''_{a}])}(\widehat{f})$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_{a}])}(\widehat{f}') = \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}'_{a}])}(\widehat{f}')$  and therefore deduce  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_{a}])}(\widehat{f}) \not\subseteq \emptyset$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}'_{a}])}(\widehat{f}') \not\subseteq \emptyset$ . But then  $(\widehat{assign}(\widehat{\rho}, x, \widehat{v}), \widehat{h} \cdot ch?\widehat{v}, \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_{a}])}(\widehat{f})) \subseteq \widehat{\mathcal{X}}(\ell)$  and  $(\widehat{assign}(\widehat{\rho}', x, \widehat{v}'), \widehat{h}' \cdot ch?\widehat{v}', \widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}'_{a}])}(\widehat{f}')) \subseteq \widehat{\mathcal{X}}'(\ell)$  by our assumption. Hence

$$\begin{split} \widehat{assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{v}'') & \doteq \widehat{assign}(\widehat{\rho}, x, \widehat{v}'') \doteq \widehat{assign}(\widehat{\rho}, x, \widehat{v}) \\ \widehat{assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{v}'') & \doteq \widehat{assign}(\widehat{\rho}', x, \widehat{v}'') \doteq \widehat{assign}(\widehat{\rho}', x, \widehat{v}') \\ \widehat{(h \& h')} \cdot ch? \widehat{v}'' & \sqsubseteq \widehat{h} \cdot ch? \widehat{v}'' & \sqsubseteq \widehat{h} \cdot ch? \widehat{v} \\ \widehat{(h \& h')} \cdot ch? \widehat{v}'' & \sqsubseteq \widehat{h}' \cdot ch? \widehat{v}'' & \sqsubseteq \widehat{h}' \cdot ch? \widehat{v}' \\ and since \\ \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}') &= \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}) \& \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}') \\ we furthermore have \\ \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}') &\subseteq \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}) &= \widehat{\mathcal{D}_{repr([ch! \widehat{v}'_{a}])}(\widehat{f}) \\ \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}') &\subseteq \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}') &= \widehat{\mathcal{D}_{repr([ch! \widehat{v}'_{a}])}(\widehat{f}) \\ \hline{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}') &\subseteq \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}') &= \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f}') \\ \hline{(assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{v}''), (\widehat{h} \& \widehat{h}') \cdot ch? \widehat{v}'', \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}')) &\subseteq \widehat{\mathcal{X}'}(\ell) \\ and hence by the definition of the greatest lower bound: \\ (\widehat{assign}(\widehat{\rho} \sqcap \widehat{\rho}', x, \widehat{v}''), (\widehat{h} \& \widehat{h}') \cdot ch? \widehat{v}'', \widehat{\mathcal{D}_{repr([ch! \widehat{v}''_{a}])}(\widehat{f} \& \widehat{f}')) &\subseteq \widehat{\mathcal{X}'}(\ell) \\ &= (\widehat{\mathcal{X} \sqcap \widehat{\mathcal{X}'})(\ell) \\ \hline (\widehat{\mathcal{X} \sqcap \widehat{\mathcal{X}'}})(\ell) \\ \hline (\widehat{\mathcal{X} \upharpoonright \widehat{\mathcal{X}'}})(\ell) \\ \hline (\widehat{\mathcal{X} \upharpoonright \widehat{\mathcal{X}'})(\ell) \\ \hline (\widehat{\mathcal{X} \upharpoonright \widehat{\mathcal{X}'}})(\ell) \\ \hline (\widehat{\mathcal{X} \rightthreetimes \widehat{\mathcal{X}'}})(\ell) \\ \hline (\widehat{\mathcal{X} \rightthreetimes \widehat{\mathcal{X}'}})(\ell) \\ \hline (\widehat{\mathcal{X} \rightthreetimes \widehat{\mathcal{X}'}})($$

case  $ch!^{\ell}e$ : By assumption

- If  $ch?\hat{v} = \widehat{project}([ch?\hat{v}_a])$  and  $\widehat{v} \sqcap \widehat{v}' \neq \bot$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}_a])}(\widehat{f}) \not\subseteq \emptyset$  then  $(\widehat{\rho}, \widehat{h} \cdot ch!(\widehat{v} \sqcap \widehat{v}'), \widehat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}_a])}(\widehat{f})) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  for all  $[ch?\hat{v}_a] \in \widehat{range}(\widehat{f})$  where  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$  and  $\widehat{v}' = \widehat{\mathcal{A}}(e, \widehat{\rho})$  and
- If  $ch?\hat{v}'' = \widehat{project}([ch?\hat{v}'_a])$  and  $\hat{v}'' \sqcap \hat{v}''' \neq \bot$  and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}'_a])}(\hat{f}') \not \vDash \emptyset$  then  $(\hat{\rho}', \hat{h}' \cdot ch!(\hat{v}'' \sqcap \hat{v}''), \widehat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}'_a])}(\hat{f}'))$  $\sqsubseteq \widehat{\mathcal{X}'}(\ell)$  for all  $[ch?\hat{v}'_a] \in \widehat{range}(\hat{f}')$  where  $(\hat{\rho}', \hat{h}', \hat{f}') = \widehat{\mathcal{E}'}(\ell)$  and  $\hat{v}''' = \widehat{\mathcal{A}}(e, \hat{\rho}')$

By definition  $(\widehat{\mathcal{E}} \sqcap \widehat{\mathcal{E}}')(\ell) = (\widehat{\rho} \sqcap \widehat{\rho}', \widehat{h} \& \widehat{h}', \widehat{f} \& \widehat{f}')$ . By monotonicity of  $\widehat{\mathcal{A}}$  we have  $\widehat{v}_{\widehat{\mathcal{A}}} = \widehat{\mathcal{A}}(e, \widehat{\rho} \sqcap \widehat{\rho}') \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho}) =$  $\widehat{v}'$  and  $\widehat{v}_{\widehat{\mathcal{A}}} \sqsubseteq \widehat{\mathcal{A}}(e, \widehat{\rho}') = \widehat{v}'''$ . Let  $[ch?\widehat{v}''_a] \in \widehat{range}(\widehat{f} \& \widehat{f}') =$  $\widehat{overlay}(\widehat{range}(\widehat{f}), \widehat{range}(\widehat{f}'))$  be given. By definition of the lower bound computed by  $\widehat{overlay}$  and the refinement order of partitions, this means that there exists equivalence classes  $[ch?\widehat{v}_a]$  and  $[ch?\widehat{v}'_a]$  such that

$$\begin{split} [ch?\hat{v}_a''] &\sqsubseteq [ch?\hat{v}_a] \in \widehat{range}(\widehat{f}) \\ [ch?\hat{v}_a''] &\sqsubseteq [ch?\hat{v}_a'] \in \widehat{range}(\widehat{f}') \end{split}$$

under a set-of-atoms ordering. These equivalence classes are unique, as no two equivalence classes of a partition overlap.

Assume  $ch?\hat{v}_p = project([ch?\hat{v}''_a])$  and  $\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}} \neq \bot$  and  $\hat{\mathcal{D}}_{repr([ch?\hat{v}''_a])}(\hat{f} \& \hat{f}') = \hat{\mathcal{D}}_{repr([ch?\hat{v}''_a])}(\hat{f}) \& \hat{\mathcal{D}}_{repr([ch?\hat{v}''_a])}(\hat{f}')$   $\not{\subseteq} \emptyset$ . By monotonicity of project we get  $ch?\hat{v}_p \sqsubseteq project([ch?\hat{v}'_a])$   $= ch?\hat{v}$  and  $ch?\hat{v}_p \sqsubseteq project([ch?\hat{v}'_a]) = ch?\hat{v}''$  and therefore both  $\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}} \sqsubseteq \hat{v} \sqcap \hat{v}' \neq \bot$  and  $\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}} \sqsubseteq \hat{v}'' \sqcap \hat{v}''' \neq \bot$ . Furthermore  $\hat{\mathcal{D}}_{repr([ch?\hat{v}'_a])}(\hat{f}) \not{\subseteq} \emptyset$  and  $\hat{\mathcal{D}}_{repr([ch?\hat{v}'_a])}(\hat{f}') \not{\subseteq}$   $\emptyset$ . and therefore  $\hat{\mathcal{D}}_{repr([ch?\hat{v}'_a])}(\hat{f}) \not{\subseteq} \emptyset$  and  $\hat{\mathcal{D}}_{repr([ch?\hat{v}'_a])}(\hat{f}')$   $\not{\subseteq} \emptyset$  by the definition of the partition being computed. As a consequence  $(\hat{\rho}, \hat{h} \cdot ch!(\hat{v} \sqcap \hat{v}'), \hat{\mathcal{D}}_{repr([ch?\hat{v}_a])}(\hat{f})) \sqsubseteq \hat{\mathcal{X}}(\ell)$  and  $(\hat{\rho}', \hat{h}' \cdot ch!(\hat{v}'' \sqcap \hat{v}'''), \hat{\mathcal{D}}_{repr([ch?\hat{v}_a])}(\hat{f}')) \sqsubseteq \hat{\mathcal{X}}'(\ell)$ . But then  $(\hat{h} \& \hat{h}') \cdot ch!(\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}}) \sqsubseteq \hat{h} \cdot ch!(\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}}) \subseteq \hat{h} \cdot ch!(\hat{v}'' \sqcap \hat{v}''')$  $(\hat{h} \& \hat{h}') \cdot ch!(\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}}) \subseteq \hat{h}' \cdot ch!(\hat{v}_p \sqcap \hat{v}_{\hat{\mathcal{A}}}) \subseteq \hat{h} \cdot ch!(\hat{v}'' \sqcap \hat{v}''')$  and furthermore 
$$\begin{split} \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f} \& \widehat{f}') &= \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f}) \& \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f}') \\ & \sqsubseteq \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f}) &= \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}])}(\widehat{f}) \\ \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f} \& \widehat{f}') &\sqsubset \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f}') &= \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}])}(\widehat{f}') \\ and therefore \\ (\widehat{\rho} \sqcap \widehat{\rho}', (\widehat{h} \& \widehat{h}') \cdot ch!(\widehat{v}_{p} \sqcap \widehat{v}_{\widehat{\mathcal{A}}}), \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f} \& \widehat{f}')) &\sqsubseteq \widehat{\mathcal{X}}(\ell) \\ (\widehat{\rho} \sqcap \widehat{\rho}', (\widehat{h} \& \widehat{h}') \cdot ch!(\widehat{v}_{p} \sqcap \widehat{v}_{\widehat{\mathcal{A}}}), \widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}''])}(\widehat{f} \& \widehat{f}')) &\sqsubseteq \widehat{\mathcal{X}}'(\ell) \end{split}$$

and hence 
$$(\hat{a} \div \hat{a}') (\hat{a} \bullet \hat{a}')$$

$$\begin{aligned} (\widehat{\rho} \stackrel{.}{\sqcap} \widehat{\rho}', (\widehat{h} \And \widehat{h}') \cdot ch! (\widehat{v}_p \sqcap \widehat{v}_{\widehat{\mathcal{A}}}), \widehat{\mathcal{D}}_{\widehat{repr}([ch^? \widehat{v}''_a])} (\widehat{f} \And \widehat{f}')) \\ & \sqsubseteq (\widehat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell) \end{aligned}$$

**case** stop<sup> $\ell$ </sup>: By assumption  $(\perp, \hat{h}, \hat{f}) \sqsubseteq \hat{\mathcal{X}}(\ell)$  and  $(\perp, \hat{h}', \hat{f}') \sqsubseteq \widehat{\mathcal{X}'}(\ell)$  where  $(\hat{\rho}, \hat{h}, \hat{f}) = \hat{\mathcal{E}}(\ell)$  and  $(\hat{\rho}', \hat{h}', \hat{f}') = \hat{\mathcal{E}}'(\ell)$ . But then  $(\perp, \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq \hat{\mathcal{X}}(\ell)$  and  $(\perp, \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq \widehat{\mathcal{X}'}(\ell)$ . As a consequence  $(\perp, \hat{h} \& \hat{h}', \hat{f} \& \hat{f}') \sqsubseteq (\hat{\mathcal{X}} \sqcap \widehat{\mathcal{X}'})(\ell)$  where  $(\hat{\mathcal{E}} \sqcap \hat{\mathcal{E}}')(\ell) = (\hat{\rho} \sqcap \hat{\rho}', \hat{h} \& \hat{h}', \hat{f} \& \hat{f}')$ .

# 

# **B.** Soundness proofs

**B.1**  $\alpha_{st}, \gamma_{st}$  is a Galois insertion

Proof.

 $\begin{array}{l} \alpha_{st} \text{ monotone: Let } S \subseteq S' \text{ be given. If } S = \emptyset \text{ then } \alpha_{st}(S) \stackrel{.}{\sqsubseteq} \alpha_{st}(S') \\ \text{ for any } S'. \text{ If } S \neq \emptyset \text{ then } S' \neq \emptyset \text{ by assumption. Hence} \end{array}$ 

 $\gamma_{st}$  monotone: Let  $\widehat{\rho} \stackrel{:}{\sqsubseteq} \widehat{\rho'}$  be given. If  $\widehat{\rho} = \bot$  then  $\gamma_{st}(\widehat{\rho}) = \emptyset \subseteq \gamma_{st}(\widehat{\rho'})$  for any  $\widehat{\rho'}$ . If  $\widehat{\rho} \neq \bot$  then  $\widehat{\rho'} \neq \bot$  by assumption. Hence

$$\begin{split} \gamma_{st}(\widehat{\rho}) \\ &= \{\rho \mid \forall x. \ \rho(x) \text{ undefined } \lor \ \rho(x) \in \gamma_v(\widehat{\rho}(x))\} \\ &\quad (by \text{ def. of } \gamma_{st}) \\ &\subseteq \{\rho \mid \forall x. \ \rho(x) \text{ undefined } \lor \ \rho(x) \in \gamma_v(\widehat{\rho}'(x))\} \\ &\quad (by \text{ mononicity of } \gamma_v) \\ &= \gamma_{st}(\widehat{\rho}) \\ \end{split}$$

 $\gamma_{st} \circ \alpha_{st}$  extensive: Let S be given. If  $S = \emptyset$  then  $\gamma_{st}(\alpha_{st}(S)) = \gamma_{st}(\perp) = \emptyset$ . If  $S \neq \emptyset$  then

$$S \subseteq \{\rho \mid \forall x. \ \rho(x) \text{ undefined } \lor \ \rho(x) \in \{\rho(x) \mid \rho \in S \\ \land \ \rho(x) \text{ defined}\}\} \\ (upward judgement) \\ \subseteq \{\rho \mid \forall x. \ \rho(x) \text{ undefined } \lor \ \rho(x) \in \gamma_v(\alpha_v(\{\rho(x) \mid \rho \in S \\ \land \ \rho(x) \text{ defined}\}))\} \\ (\gamma_v \circ \alpha_v \text{ extensive}) \\ = \gamma_{st}(\lambda x. \ \alpha_v(\{\rho(x) \mid \rho \in S \land \ \rho(x) \text{ defined}\}))$$

 $(by \text{ def. of } \gamma_{st})$  $= \gamma_{st}(\alpha_{st}(S)) \qquad (by \text{ def. of } \alpha_{st})$ 

 $\alpha_{st} \circ \gamma_{st}$  identity: Let  $\hat{\rho}$  be given. If  $\hat{\rho} = \bot$  then  $\alpha_{st}(\gamma_{st}(\hat{\rho})) = \alpha_{st}(\emptyset) = \bot$ . If  $\hat{\rho} \neq \bot$  then

$$\begin{aligned} \alpha_{st}(\gamma_{st}(\widehat{\rho})) \\ &= \alpha_{st}(\{\rho \mid \forall x. \ \rho(x) \text{ undefined } \lor \ \rho(x) \in \gamma_v(\widehat{\rho}(x))\}) \\ &\quad (by \text{ def. of } \gamma_{st}) \end{aligned}$$

$$&= \lambda x. \ \alpha_v(\{\rho(x) \mid \rho \in \{\rho \mid \forall x'. \ \rho(x') \text{ undefined} \\ &\quad \lor \ \rho(x') \in \gamma_v(\widehat{\rho}(x'))\} \land \ \rho(x) \text{ defined}\}) \\ &\quad (by \text{ def. of } \alpha_{st}) \end{aligned}$$

$$&= \lambda x. \ \alpha_v(\gamma_v(\widehat{\rho}(x))) \qquad \qquad (simplify) \\ &= \lambda x. \ \widehat{\rho}(x) \qquad \qquad (\alpha_v \circ \gamma_v \text{ identity}) \end{aligned}$$

# **B.2** Soundness of $\widehat{\mathcal{A}}$

*Proof.* By structural induction on e. Let  $e \in E, \hat{\rho} \in \widehat{Store}$  be given.

case n:

$$\begin{aligned} \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} n \Downarrow v\}) \\ &= \alpha_v(\{n \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} n \Downarrow n\}) \quad \text{(by rule LIT)} \\ &= \alpha_v(\{n \mid \rho \in \gamma_{st}(\widehat{\rho})\}) \quad \text{(simplify)} \\ &\sqsubseteq \alpha_v(\{n\}) \quad (\alpha_v \text{ monotone}) \\ &= \widehat{\mathcal{A}}(n, \widehat{\rho}) \quad \text{(by def. of } \widehat{\mathcal{A}}) \end{aligned}$$

case x:

$$\begin{aligned} \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} x \Downarrow v\}) \\ &= \alpha_v(\{\rho(x) \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} x \Downarrow \rho(x)\}) \\ & \text{(by rule VAR)} \\ &= \alpha_v(\{\rho(x) \mid \rho \in \gamma_{st}(\widehat{\rho})\}) \\ &= \begin{cases} \alpha_v(\emptyset) & \widehat{\rho} = \bot \\ \alpha_v(\gamma_v(\widehat{\rho}(x))) & \widehat{\rho} \neq \bot \end{cases} \\ & \text{(case analysis)} \\ &= \begin{cases} \bot & \widehat{\rho} = \bot \\ \widehat{\rho}(x) & \widehat{\rho} \neq \bot \end{cases} \\ & \text{(Galois insertion, strict } \gamma_v) \\ &= \widehat{\mathcal{A}}(x, \widehat{\rho}) \\ & \text{(by def. of } \widehat{\mathcal{A}}) \end{aligned}$$

case ?:

$\alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} ? \Downarrow v\})$	
$= \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} ? \Downarrow v\})$	(by rule ANY)
$= \alpha_v(\{v \mid \rho \in \gamma_{st}(\widehat{\rho})\})$	(simplify)
$\subseteq \top$	(by def. of $\top$ )
$=\widehat{\mathcal{A}}(?,\widehat{ ho})$	(by def. of $\widehat{\mathcal{A}}$ )

case  $e_1 + e_2$ :

 $\alpha_{v}(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_{1} + e_{2} \Downarrow v\})$  $= \alpha_v(\{v_1 + v_2 \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_1 \Downarrow v_1 \land \rho \vdash_{\mathcal{A}} e_2 \Downarrow v_2\})$ (by rule ADD)  $\sqsubseteq \alpha_v(\{v_1+v_2 \mid v_1 \in \{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_1 \Downarrow v\} \land$  $v_2 \in \{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_2 \Downarrow v\}\})$  $(\alpha_v \text{ monotone})$  $\sqsubseteq \alpha_v(\{v_1 + v_2 \mid v_1 \in \gamma_v(\widehat{\mathcal{A}}(e_1, \widehat{\rho})) \land v_2 \in \gamma_v(\widehat{\mathcal{A}}(e_2, \widehat{\rho}))\})$ (by the IH, Galois connection)  $\sqsubset \widehat{\mathcal{A}}(e_1,\widehat{\rho}) + \widehat{\mathcal{A}}(e_2,\widehat{\rho})$ (by + soundness) $=\widehat{\mathcal{A}}(e_1+e_2,\widehat{\rho})$ (by def. of  $\widehat{\mathcal{A}}$ ) case  $e_1 - e_2$ :  $\alpha_{v}(\{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_{1} - e_{2} \Downarrow v\})$  $= \alpha_v(\{v_1 - v_2 \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_1 \Downarrow v_1 \land \rho \vdash_{\mathcal{A}} e_2 \Downarrow v_2\})$ (by rule ADD)  $\sqsubseteq \alpha_v(\{v_1 - v_2 \mid v_1 \in \{v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_1 \Downarrow v\} \land$  $v_2 \in \{ v \mid \rho \in \gamma_{st}(\widehat{\rho}) \land \rho \vdash_{\mathcal{A}} e_2 \Downarrow v \} \} )$  $(\alpha_v \text{ monotone})$  $\sqsubseteq \alpha_v(\{v_1 - v_2 \mid v_1 \in \gamma_v(\widehat{\mathcal{A}}(e_1, \widehat{\rho})) \land v_2 \in \gamma_v(\widehat{\mathcal{A}}(e_2, \widehat{\rho}))\})$ (by the IH, Galois connection)  $\sqsubseteq \widehat{\mathcal{A}}(e_1,\widehat{\rho}) \widehat{-} \widehat{\mathcal{A}}(e_2,\widehat{\rho})$ (by - soundness) $=\widehat{\mathcal{A}}(e_1-e_2,\widehat{\rho})$ (by def. of  $\widehat{\mathcal{A}}$ ) 

# **B.3** Soundness of *assign*

*Proof.* Let  $\hat{\rho}$ , x,  $\hat{v}$  be given. If  $\gamma_v(\hat{v}) = \emptyset$  or  $\gamma_{st}(\hat{\rho}) = \emptyset$  the left-hand-side is  $\bot$  and hence the result follows immediately. Therefore

assume  $\gamma_v(\hat{v}) \neq \emptyset$  and  $\gamma_{st}(\hat{\rho}) \neq \emptyset$ . To argue for pointwise inclusion, we prove inclusion for a given variable y. Let a y be given.

case y = x:

$$\begin{aligned} \alpha_v(\{\rho[x\mapsto v](y) \mid v \in \gamma_v(\widehat{v}) \land \rho \in \gamma_{st}(\widehat{\rho}) \land \rho[x\mapsto v](y) \text{ defined}\}) \\ &= \alpha_v(\{v \mid v \in \gamma_v(\widehat{v}) \land \rho \in \gamma_{st}(\widehat{\rho})\}) \qquad \text{(by } y = x) \\ &= \alpha_v(\gamma_v(\widehat{v})) \qquad (\gamma_v(\widehat{v}) \neq \emptyset, \gamma_{st}(\widehat{\rho}) \neq \emptyset) \\ &= \widehat{v} \qquad \text{(Galois insertion)} \\ &= \widehat{assign}(\widehat{\rho}, x, \widehat{v})(y) \qquad \text{(by def. of } \widehat{assign}) \end{aligned}$$

case  $y \neq x$ :

# **B.4** Soundness of generic $\widehat{true}$ and $\widehat{false}$

*Proof.* By structural induction on b. Let  $b \in B$ ,  $\hat{\rho} \in \widehat{Store}$  be given. case tt:

$$\begin{aligned} \alpha_{st}(true(\texttt{tt},\gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} \texttt{tt} \Downarrow \texttt{tt}\}) \quad (\texttt{by def. of } true) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho})\}) \quad (\texttt{by rule TRUE}) \\ &= \widehat{\rho} \quad (\texttt{Galois insertion}) \\ &= \widehat{true}(\texttt{tt},\widehat{\rho}) \quad (\texttt{by def. of } \widehat{true}) \end{aligned}$$

$lpha_{st}(\mathit{false}(\mathtt{tt},\gamma_{st}(\widehat{ ho})))$	
$= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} \mathtt{tt} \Downarrow \mathtt{ff}\})$	(by def. of <i>false</i> )
$= \alpha_{st}(\emptyset)$	(no rules apply)
$= \bot$	(by def. of $\alpha_{st}$ )
$=\widehat{false}(\mathtt{tt},\widehat{ ho})$	(by def. of $\widehat{false}$ )

) case ff:

$$\begin{aligned} \alpha_{st}(true(\texttt{ff}, \gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} \texttt{ff} \Downarrow \texttt{tt}\}) & (\text{by def. of } true) \\ &= \alpha_{st}(\emptyset) & (\text{no rules apply}) \\ &= \bot & (\text{by def. of } \alpha_{st}) \\ &= \widehat{true}(\texttt{ff}, \widehat{\rho}) & (\text{by def. of } \widehat{true}) \end{aligned}$$

$$\begin{aligned} &\alpha_{st}(false(\texttt{ff}, \gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} \texttt{ff} \Downarrow \texttt{ff}\}) & (\texttt{by def. of } false) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho})\}) & (\texttt{by rule FALSE}) \\ &= \widehat{\rho} & (\texttt{Galois insertion}) \\ &= \widehat{false}(\texttt{ff}, \widehat{\rho}) & (\texttt{by def. of } \widehat{false}) \end{aligned}$$

**case** 
$$x_1 < x_2$$
:

$$\begin{aligned} \alpha_{st}(true(x_1 < x_2, \gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} x_1 < x_2 \Downarrow \mathsf{tt}\}) \text{ (by def. of } true) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho(x_1) < \rho(x_2)\}) \\ & \text{ (by rule LESSTHAN1)} \\ & \vdots \\ \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \widehat{\rho}(x_1) \neq \bot \land \widehat{\rho}(x_2) \neq \bot\}) \\ & (\gamma_v \text{ strict, } \alpha_{st} \text{ monotone}) \\ &= \begin{cases} \bot & \widehat{\rho}(x_1) = \bot \lor \widehat{\rho}(x_2) = \bot \\ \widehat{\rho} & \text{ otherwise} \end{cases} \end{aligned}$$

# **B.5** Soundness of $\widehat{true}$ and $\widehat{false}$ for intervals

*Proof.* By structural induction on b. Let  $b \in B$ ,  $\hat{\rho} \in Store$  be given. The cases for tt and ff are identical to those for parity and so is the proof.

**case**  $x_1 < x_2$ :

$$\begin{split} \alpha_{st}(true(x_{1} < x_{2}, \gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} x_{1} < x_{2} \Downarrow \texttt{tt}\}) \text{ (by def. of } true) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho(x_{1}) < \rho(x_{2})\}) \\ & \text{(by rule LESSTHAN1)} \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid l_{1} \leq v_{1} = \rho(x_{1}) \leq u_{1} \land \\ & l_{2} \leq v_{2} = \rho(x_{2}) \leq u_{2} \land v_{1} < v_{2}\}) \\ & \text{(by def. of } \gamma_{st}) \\ & \stackrel{\bot}{\sqsubseteq} \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid l_{1} \leq v_{1} = \rho(x_{1}) \leq (\min u_{1}(u_{2} - 1)) \land \\ & (\max(l_{1} + 1)l_{2}) \leq v_{2} = \rho(x_{2}) \leq u_{2}\}) \\ & \text{(by def. of } <, \leq) \\ & \stackrel{\bot}{\sqsubseteq} \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}[x_{1} \mapsto [l_{1}; u'_{1}], x_{2} \mapsto [l'_{2}; u_{2}]]) \mid \widehat{\rho}(x_{i}) = [l_{i}; u_{i}] \\ & \wedge u'_{1} = \min u_{1}(u_{2} - 1) \land l'_{2} = \max(l_{1} + 1)l_{2}\}) \\ & (\alpha_{st} \text{ monotone, } \rho(x_{i}) \text{ pot. undefined}) \\ & = \begin{cases} \\ & \stackrel{\bot}{\rho}[x_{1} \mapsto [l_{1}; u'_{1}], x_{2} \mapsto [l'_{2}; u_{2}]] & \widehat{\rho}(x_{i}) = [l_{i}; u_{i}] \land \\ & u'_{1} = \min u_{1}(u_{2} - 1) \land \\ & l'_{2} = \max(l_{1} + 1)l_{2} \\ & \text{(by def. of } \alpha_{st}) \end{cases} \end{split}$$

$$\begin{aligned} \alpha_{st}(false(x_{1} < x_{2}, \gamma_{st}(\widehat{\rho}))) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho \vdash_{\mathcal{B}} x_{1} < x_{2} \Downarrow \mathsf{ff}\}) \\ & (by \text{ def. of } false) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid \rho(x_{1}) \ge \rho(x_{2})\}) \\ & (by \text{ rule LESSTHAN2, Galois insertion}) \\ &= \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid l_{1} \le v_{1} = \rho(x_{1}) \le u_{1} \land \\ l_{2} \le v_{2} = \rho(x_{2}) \le u_{2} \land v_{1} \ge v_{2}\}) \\ & (by \text{ def. of } \gamma_{st}) \end{aligned}$$

$$\dot{\sqsubseteq} \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}) \mid (maxl_{1}l_{2}) \le v_{1} = \rho(x_{1}) \le u_{1} \land \\ l_{2} \le v_{2} = \rho(x_{2}) \le (minu_{1}u_{2})\}) \\ & (by \text{ def. of } <, \le) \end{aligned}$$

$$\dot{\sqsubseteq} \alpha_{st}(\{\rho \in \gamma_{st}(\widehat{\rho}[x_{1} \mapsto [l'_{1}; u_{1}], x_{2} \mapsto [l_{2}; u'_{2}]]) \mid \widehat{\rho}(x_{i}) = [l_{i}; u_{i}] \land \\ l'_{1} = \max l_{1}l_{2} \land u'_{2} = \min u_{1}u_{2}\}) \\ & (\alpha_{st} \text{ monotone, } \rho(x_{i}) \text{ pot. undefined}) \end{aligned}$$

$$= \begin{cases} \downarrow \\ \widehat{\rho}[x_{1} \mapsto [l'_{1}; u_{1}], x_{2} \mapsto [l_{2}; u'_{2}]] & \widehat{\rho}(x_{i}) = [l_{i}; u_{i}] \land \\ l'_{1} = \max l_{1}l_{2} \land \\ u'_{2} = \min u_{1}u_{2} \land \\ u'_{2} = \min u_{1}u_{2} \end{cases} \\ & (by \text{ def. of } \alpha_{st}) \end{cases}$$

# **B.6** Soundness of instrumented semantics

Proof. By induction over the length of the trace.

$$n = 1: \text{ If } c_1 \parallel \overline{c}_1 \text{ for some initial state } c_1 \parallel \overline{c}_1 = \langle s_1^{\ell_1}, \rho_1 \rangle \parallel \\ \frac{\langle \overline{s_1}^{\ell_1}, \overline{\rho}_1 \rangle}{ic_1 = \langle s_1^{\ell_1}, \rho_1, \epsilon \rangle} \parallel \langle \overline{s_1}^{\ell_1}, \overline{\rho}_1, \epsilon \rangle \text{ and by definition } \langle s_1^{\ell_1}, \rho_1 \rangle, \epsilon = \\ proj(\langle s_1^{\overline{\ell}_1}, \rho_1, \epsilon \rangle) \land \langle \overline{s_1}^{\ell_1}, \overline{\rho}_1 \rangle, \epsilon = proj(\langle \overline{s_1}^{\ell_1}, \overline{\rho}_1, \epsilon \rangle) \text{ and } \\ \text{clearly } \epsilon = \epsilon \land \epsilon = \epsilon. \end{cases}$$

$$n = k + 1: \text{ Let a trace } c_1 \parallel \overline{c}_1 \xrightarrow{\alpha_1, \beta_1} c_2 \parallel \overline{c}_2 \xrightarrow{\alpha_2, \beta_2} \cdots \xrightarrow{\alpha_k, \beta_k} \\ c_{k+1} \parallel \overline{c}_{k+1} \text{ from some initial state } c_1 \parallel \overline{c}_1 = \langle s_1^{\ell_1}, \rho_1 \rangle \parallel \\ \langle \overline{s_1}^{\ell_1}, \overline{\rho}_1 \rangle \text{ be given. By the induction hypothesis there exists an instrumented trace } ic_1 \parallel \overline{ic_1} \Longrightarrow ic_2 \parallel \overline{ic_2} \Longrightarrow \ldots \Longrightarrow ic_k \parallel \\ \overline{ic_k} \text{ such that}$$

$$c_i, h_i = proj(ic_i) \land \overline{c}_i, \overline{h}_i = proj(\overline{ic}_i) \qquad i \in \{1, \dots, k\}$$
$$\land \alpha_1 \alpha_2 \dots \alpha_{k-1} = h_k \land \beta_1 \beta_2 \dots \beta_{k-1} = \overline{h}_k$$

We need to show that there exists  $ic_{k+1} \parallel \overline{ic}_{k+1}$  such that  $ic_k \parallel \overline{ic}_k \Longrightarrow ic_{k+1} \parallel \overline{ic}_{k+1}$  and

$$c_{k+1}, h_{k+1} = proj(ic_{k+1}) \land \overline{c}_{k+1}, \overline{h}_{k+1} = proj(\overline{ic}_{k+1}) \land \alpha_1 \alpha_2 \dots \alpha_k = h_{k+1} \land \beta_1 \beta_2 \dots \beta_k = \overline{h}_{k+1}$$

We proceed by case analysis on the last step of the trace  $c_k \parallel \overline{c_k} \stackrel{\alpha_k, \beta_k}{\Longrightarrow} c_{k+1} \parallel \overline{c_{k+1}}$ .

**case SysLeft:** Then  $\langle s_k^{\ell_k}, \rho_k \rangle \parallel \overline{c}_k \stackrel{\tau, \epsilon}{\Longrightarrow} c_{k+1} \parallel \overline{c}_k$  for some  $\langle s_k^{\ell_k}, \rho_k \rangle \stackrel{\tau}{\longrightarrow} c_{k+1}$ . By Lemma 7.5  $\langle s_k^{\ell_k}, \rho_k, h_k \rangle \stackrel{\tau}{\longrightarrow} ic_{k+1}$  such that  $c_{k+1}, (h_k \cdot \tau) = c_{k+1}, h_{k+1} = proj(ic_{k+1})$ . Hence by rule ISYSLEFT we have  $\langle s_k^{\ell_k}, \rho_k, h_k \rangle \parallel ic_k \implies ic_{k+1} \parallel ic_k$ . By the induction hypothesis  $\overline{c}_{k+1}, \overline{h}_{k+1} = \overline{c}_k, \overline{h}_k = proj(ic_k) = proj(ic_{k+1})$ . Since  $\alpha_1 \alpha_2 \dots \alpha_{k-1} = h_k \land \beta_1 \beta_2 \dots \beta_{k-1} = \overline{h}_k$  and  $\alpha_k = \tau$  and  $\beta_k = \epsilon$  then  $\alpha_1 \alpha_2 \dots \alpha_{k-1} \alpha_k = h_k \cdot \tau = h_{k+1}$  and  $\beta_1 \beta_2 \dots \beta_{k-1} \beta_k = \overline{h}_k \cdot \epsilon = \overline{h}_k = \overline{h}_{k+1}$ .

case SYSRIGHT: The argument is symmetric to that of SYSLEFT.

**case SysWR:** Then  $\langle s_k^{\ell_k}, \rho_k \rangle \parallel \langle \overline{s}_k^{\overline{\ell}_k}, \overline{\rho}_k \rangle \stackrel{ch!v.ch?v}{\Longrightarrow} c_{k+1} \parallel \overline{c}_{k+1}$  and  $\langle s_k^{\ell_k}, \rho_k \rangle \stackrel{ch!v}{\Longrightarrow} c_{k+1}$  and  $\langle \overline{s}_k^{\ell_k}, \overline{\rho}_k \rangle \stackrel{ch?v}{\longrightarrow} \overline{c}_{k+1}$ . Hence by two applications of Lemma 7.5 we get  $\langle s_k^{\ell_k}, \rho_k, h_k \rangle$  $ic_{k+1}$  and  $\langle \overline{s}_k^{\overline{\ell}_k}, \overline{\rho}_k, \overline{h}_k \rangle \xrightarrow{ch?v} \overline{ic}_{k+1}$ . and  $c_{k+1}, (h_k + 1)$  $ch!v) = proj(ic_{k+1})$  and  $\overline{c}_{k+1}, (\overline{h}_k \cdot ch?v) = proj(\overline{ic}_{k+1}).$ By rule ISYSWR we therefore have  $\langle s_k^{\ell_k}, \rho_k, h_k \rangle \parallel \langle \overline{s}_k^{\overline{\ell}_k}, \overline{\rho}_k, \overline{h}_k \rangle \implies c_{k+1} \parallel \overline{c}_{k+1}$ . Furthermore by the induction hypothesis we have that  $c_i, h_i = proj(ic_i) \wedge \overline{c}_i, \overline{h}_i =$  $proj(ic_i)$  for all  $i \in \{1, \ldots, k\}$  and  $\alpha_1 \alpha_2 \ldots \alpha_{k-1} =$  $h_k \wedge \beta_1 \beta_2 \dots \beta_{k-1} = \overline{h}_k$  and therefore  $\alpha_1 \alpha_2 \dots \alpha_{k-1} ch! v$  $= h_k \cdot ch! v = h_{k+1} \wedge \beta_1 \beta_2 \dots \beta_{k-1} ch? v = \overline{h_k} \cdot ch? v =$  $\overline{h}_{k+1}$ .

case SYSRW: The argument is symmetric to that of SYSWR.

## **B.7** Process step soundness

*Proof.* By structural induction on  $s^{\ell}$ . Let  $s^{\ell}$ ,  $\rho$ , and h be given. Assume  $\langle s^{\ell}, \rho \rangle, h = proj(\langle s^{\ell}, \rho, h \rangle)$  and  $\langle s^{\ell}, \rho \rangle \xrightarrow{\alpha} c$ .

- **case** skip<sup> $\ell$ </sup>:  $\langle \text{skip}^{\ell}, \rho \rangle, h = proj(\langle \text{skip}^{\ell}, \rho, h \rangle) \text{ and } \langle \text{skip}^{\ell}, \rho \rangle \xrightarrow{\tau}$ c by rule SKIP. By rule ISKIP  $\langle \text{skip}^{\ell}, \rho, h \rangle \xrightarrow{\tau} \langle \rho, h \cdot \tau \rangle$  and  $\rho, (h \cdot \tau) = proj(\langle \rho, h \cdot \tau \rangle).$
- case  $x := {}^{\ell}e: \langle x := {}^{\ell}e, \rho \rangle, h = proj(\langle x := {}^{\ell}e, \rho, h \rangle) \text{ and } \langle x := {}^{\ell}e, \rho \rangle \xrightarrow{\tau} \lambda$  $\rho[x \mapsto v]$  by rule ASSIGN for some v where  $e \vdash_{\mathcal{A}} \rho \Downarrow v$ . But then by rule IASSIGN  $\langle x :=^{\ell} e, \rho, h \rangle \xrightarrow{\tau} \langle \rho[x \mapsto v], h \cdot \tau \rangle$ and  $\rho[x \mapsto v], (h \cdot \tau) = proj(\langle \rho[x \mapsto v], h \cdot \tau \rangle).$
- $\stackrel{\alpha}{\longrightarrow} c$  by either rule SEQ1 or by rule SEQ2.
  - subcase SEQ1: In this case we have that  $\langle s_1^{\ell_1}, \rho \rangle \xrightarrow{\alpha} \langle s_3^{\ell_3}, \rho' \rangle$ and  $\langle s_1^{\ell_1}; s_2^{\ell_2}, \rho \rangle \xrightarrow{\alpha} \langle s_3^{\ell_3}; s_2^{\ell_2}, \rho' \rangle$ . But then  $\langle s_1^{\ell_1}, \rho \rangle, h =$  $proj(\langle s_1^{\ell_1}, \rho, h \rangle)$  and therefore  $\langle s_1^{\ell_1}, \rho, h \rangle \xrightarrow{\alpha} \langle s_3^{\ell_3}, \rho', h \cdot \alpha \rangle$ by the induction hypothesis. Hence by rule ISEQ1  $\langle s_1^{\ell_1}\,;s_2^{\ell_2},\rho,h\rangle$  $\stackrel{\alpha}{\longrightarrow} \langle s_3^{\ell_3}; s_2^{\ell_2}, \rho', h \cdot \alpha \rangle \text{ and } \langle s_3^{\ell_3}; s_2^{\ell_2}, \rho' \rangle, (h \cdot \alpha) \stackrel{\alpha}{=}$  $proj(\langle s_3^{\ell_3}; s_2^{\ell_2}, \rho', h \cdot \alpha \rangle).$
  - subcase SEQ2: In this case we have that  $\langle s_1^{\ell_1}, \rho \rangle \xrightarrow{\alpha} \rho'$ and  $\langle s_1^{\ell_1}; s_2^{\ell_2}, \rho \rangle \xrightarrow{\alpha} \langle s_2^{\ell_2}, \rho' \rangle$ . But then  $\langle s_1^{\ell_1}, \rho \rangle, h =$  $proj(\langle s_1^{\ell_1}, \rho, h \rangle)$  and therefore  $\langle s_1^{\ell_1}, \rho, h \rangle \xrightarrow{\alpha} \langle \rho', h \cdot \alpha \rangle$ by the induction hypothesis. Hence by rule ISEQ2  $\langle s_1^{\ell_1}; s_2^{\ell_2}, \rho, h \rangle$  $\stackrel{\alpha}{\longrightarrow} \langle s_2^{\ell_2}, \rho', h \cdot \alpha \rangle \text{ and } \langle s_2^{\ell_2}, \rho' \rangle, (h \cdot \alpha) = proj(\langle s_2^{\ell_2}, \rho', h \cdot \alpha \rangle).$
- case if  $b^{\ell}$  then  $s_1^{\ell_1}$  else  $s_2^{\ell_2}$ : (if  $b^{\ell}$  then  $s_1^{\ell_1}$  else  $s_2^{\ell_2}, \rho$ ),  $h = b^{\ell_1}$  $proj(\langle \texttt{if} \ b^\ell \ \texttt{then} \ s_1^{\ell_1} \ \texttt{else} \ s_2^{\ell_2}, \rho, h \rangle) \text{ and } \langle \texttt{if} \ b^\ell \ \texttt{then} \ s_1^{\ell_1} \ \texttt{else} \ s_2^{\ell_2}, \rho \rangle$  $\stackrel{\alpha}{\longrightarrow} c$  by either rule IF1 or by rule IF2.
  - $\textbf{subcase IF1: Then } b \vdash_{\mathcal{B}} \rho \Downarrow \texttt{tt} \texttt{and} \ \texttt{(if} \ b^\ell \texttt{ then } s_1^{\ell_1} \texttt{ else } s_2^{\ell_2}, \rho \texttt{)}$  $\begin{array}{l} \stackrel{\tau}{\longrightarrow} \langle s_1^{\ell_1}, \rho \rangle. \, \text{But then } \langle \text{if } b^\ell \text{ then } s_1^{\ell_1} \text{ else } s_2^{\ell_2}, \rho, h \rangle \stackrel{\tau}{\longrightarrow} \\ \langle s_1^{\ell_1}, \rho, h \cdot \tau \rangle \text{ by rule IIF1 and } \langle s_1^{\ell_1}, \rho \rangle, (h \cdot \tau) = proj(\langle s_1^{\ell_1}, \rho, h \cdot \tau \rangle). \end{array}$  $\textbf{subcase IF2: Then } b \vdash_{\mathcal{B}} \rho \Downarrow \texttt{ff and } \langle \texttt{if } b^\ell \texttt{ then } s_1^{\ell_1} \texttt{ else } s_2^{\ell_2}, \rho \rangle$  $\stackrel{\tau}{\longrightarrow} \langle s_2^{\ell_2}, \rho \rangle. \text{ But then } \langle \texttt{if } b^\ell \texttt{ then } s_1^{\ell_1} \texttt{ else } s_2^{\ell_2}, \rho, h \rangle \stackrel{\tau}{\longrightarrow}$
- $\langle s_2^{\ell_2}, \rho, h \cdot \tau \rangle \text{ by rule IIF2 and } \langle s_2^{\ell_2}, \rho \rangle, (h \cdot \tau) = proj(\langle s_2^{\ell_2}, \rho, h \cdot \tau \rangle)?$ case while  $b^{\ell}$  do  $s_1^{\ell_1}$  end:
  - $\langle \texttt{while} \ b^\ell \ \texttt{do} \ s_1^{\bar{\ell_1}} \ \texttt{end}, \rho \rangle, h = proj(\langle \texttt{while} \ b^\ell \ \texttt{do} \ s_1^{\ell_1} \ \texttt{end}, \rho, h \rangle)$ and (while  $b^\ell$  do  $s_1^{\ell_1}$  end,  $\rho$ )  $\xrightarrow{\tau} c$  by either rule WHILE1 or by rule WHILE2.
  - **subcase WHILE1:** Then  $b \vdash_{\mathcal{B}} \rho \Downarrow \text{tt}$  and  $\langle \text{while } b^{\ell} \text{ do } s_1^{\ell_1} \text{ end}, \rho \rangle$  $\stackrel{\tau}{\longrightarrow} \ \langle s_1^{\ell_1} \, ; \texttt{while} \ b^\ell \ \texttt{do} \ s_1^{\ell_1} \ \texttt{end}, \rho \rangle. \ \text{But then we have}$ (while  $b^{\ell}$  do  $s_1^{\ell_1}$  end,  $\rho, h$ )  $\xrightarrow{\tau} \langle s_1^{\ell_1}$ ; while  $b^{\ell}$  do  $s_1^{\ell_1}$  end,  $\rho, h; \tau$ ) Part 2: Let  $\alpha \neq \tau$  be given by rule IWHILE1 and  $\langle s_1^{\ell_1}; \text{while } b^\ell \text{ do } s_1^{\ell_1} \text{ end, } \rho \rangle, (h \cdot \tau) = proj(\langle s_1^{\ell_1}; \text{while } b^\ell \text{ do } s_1^{\ell_1} \text{ end, } \rho, h \cdot \tau \rangle).$

- **subcase WHILE2:** Then  $b \vdash_{\mathcal{B}} \rho \Downarrow \text{ ff and } \langle \text{while } b^{\ell} \text{ do } s_1^{\ell_1} \text{ end}, \rho \rangle$  $\stackrel{\tau}{\longrightarrow}\rho. \text{ But then } \langle \texttt{while } b^\ell \text{ do } s_1^{\ell_1} \text{ end}, \rho, h \rangle \stackrel{\tau}{\longrightarrow} \langle \rho, h \cdot \tau \rangle$ by rule IWHILE2 and  $\rho$ ,  $(h \cdot \tau) = proj(\langle \rho, h \cdot \tau \rangle)$ .
- $\xrightarrow{ch!v} \operatorname{case} s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2} \colon \langle s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}, \rho \rangle, h = \operatorname{proj}(\langle s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}, \rho, h \rangle)$ and  $\langle s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}, \rho \rangle \xrightarrow{\alpha} c$  by either rule CHOICE1 or by rule CHOICE2.
  - subcase Choice1: In this case we have that  $\langle s_1^{\ell_1}, \rho \rangle \xrightarrow{\alpha}$  $c_1 \text{ and } \langle s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}, \rho \rangle \xrightarrow{\alpha} c_1. \text{ But then } \langle s_1^{\ell_1}, \rho \rangle, h = proj(\langle s_1^{\ell_1}, \rho, h \rangle) \text{ and therefore } \langle s_1^{\ell_1}, \rho, h \rangle \xrightarrow{\alpha} ic_1 \text{ and } c_1$  $c_1, (h \cdot \alpha) = proj(ic_1)$  by the induction hypothesis. Hence  $\langle s_1^{\ell_1} \oplus^{\ell} s_2^{\ell_2}, \rho, h \rangle \xrightarrow{\alpha} ic_1$  by rule ICHOICE1.

subcase CHOICE2: Symmetric to CHOICE1.

case  $ch?^{\ell}x$ :  $\langle ch?^{\ell}x, \rho \rangle, h = proj(\langle ch?^{\ell}x, \rho, h \rangle)$  and  $\langle ch?^{\ell}x, \rho \rangle$  $\stackrel{ch?v}{\longrightarrow} \rho[x \mapsto v]$  by rule READ for some v. But then by rule 

**case**  $ch!^{\ell}e$ :  $\langle ch!^{\ell}e, \rho \rangle, h = proj(\langle ch!^{\ell}e, \rho, h \rangle)$  and  $\langle ch!^{\ell}e, \rho \rangle \xrightarrow{ch!v}$  $\rho$  by rule WRITE for some v such that  $\rho \vdash_{\mathcal{A}} e \Downarrow v$ . But then by rule IWRITE  $\langle ch!^{\ell}e, \rho, h \rangle \xrightarrow{ch!v} \langle \rho, h \cdot ch!v \rangle$  and  $\rho, (h \cdot ch!v) = proj(\langle \rho, h \cdot ch!v \rangle).$ 

**case** stop: Vacuously true as no rule matches  $\langle \texttt{stop}, \rho \rangle \xrightarrow{\alpha} c$ .

## **B.8** Decoupling a system trace

Proof. By induction on the length of the trace.

- $\mathbf{case} \ s_1^{\ell_1}; s_2^{\ell_2}: \ \langle s_1^{\ell_1}; s_2^{\ell_2}, \rho \rangle, h = proj(\langle s_1^{\ell_1}; s_2^{\ell_2}, \rho, h \rangle) \text{ and } \langle s_1^{\ell_1}; s_2^{\ell_2}, \rho \rangle \text{ case } n = 1: \text{ Clearly we can decouple } ic_1 \parallel \overline{ic_1} \text{ into two traces } ic_1 \parallel \overline{ic_2} \mid | \overline{$ and  $\overline{ic_1}$ .
  - **case** n = k + 1: Assume we are given a system trace:  $ic_1 \parallel$  $\overline{ic}_1 \Longrightarrow ic_2 \parallel \overline{ic}_2 \Longrightarrow \ldots \Longrightarrow ic_k \parallel \overline{ic}_k \Longrightarrow ic_{k+1} \parallel \overline{ic}_{k+1}.$ By the induction hypothesis we can decouple the first k states of the trace into two process traces  $ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{k-1}} ic_k$ and  $\overline{ic_1} \xrightarrow{\beta_1} \overline{ic_2} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{k-1}} \overline{ic_k}$ .

We now proceed by case analysis on the last step.

- subcase ISYSLEFT: Then  $ic_k = \langle s_1, \rho_1, h_1 \rangle \xrightarrow{\tau} ic_{k+1}$  and  $\overline{ic}_k = \overline{ic}_{k+1}$ . But then  $ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{k-1}} ic_k \xrightarrow{\tau}$  $ic_{k+1} \text{ and } \overline{ic_1} \xrightarrow{\beta_1} \overline{ic_2} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{k-1}} \overline{ic_k} \xrightarrow{\epsilon} \overline{ic_{k+1}}.$
- subcase ISYSRIGHT: The argument is symmetric to that of ISYSLEFT.
- subcase ISYSWR: Then  $ic_k = \langle s_1, \rho_1, h_1 \rangle \xrightarrow{ch!v} ic_{k+1}$  and  $\overline{ic_k} = \langle s_2, \rho_2, h_2 \rangle \xrightarrow{ch^{\circ}v} \overline{ic_{k+1}}. \text{ Therefore } ic_1 \xrightarrow{\alpha_1} ic_2 \xrightarrow{\alpha_2}$  $\dots \xrightarrow{\alpha_{k-1}} ic_k \xrightarrow{ch!v} ic_{k+1} \text{ and } \overline{ic_1} \xrightarrow{\beta_1} \overline{ic_2} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{k-1}}$  $\overline{ic}_k \xrightarrow{ch?v} \overline{ic}_{k+1}.$

subcase ISYSRW: The argument is symmetric to that of ISYSWR.

### Actions at the end of a string

*Proof.* Both parts are by structural induction on h. Let h be given. Part 1:

case  $\epsilon$ :  $|\epsilon \cdot \tau| = |\tau| = |\tau \cdot \epsilon| = |\epsilon|$ . case  $\alpha \cdot h$ :  $|(\alpha \cdot h) \cdot \tau| = |\alpha \cdot (h \cdot \tau)|$  If  $\alpha = \tau$ :  $|\alpha \cdot (h \cdot \tau)| =$  $|h \cdot \tau| = |h| = |\alpha \cdot h|$ . If  $\alpha \neq \tau$ :  $|\alpha \cdot (h \cdot \tau)| = \alpha \cdot |h \cdot \tau| =$  $\alpha \cdot |h| = |\alpha \cdot h|.$ 

case 
$$\epsilon$$
:  $|\epsilon \cdot \alpha| = |\alpha| = \alpha \cdot |\epsilon| = \alpha \cdot \epsilon = \alpha = \epsilon \cdot \alpha = |\epsilon| \cdot \alpha$ 

case  $\beta \cdot h$ :  $|(\beta \cdot h) \cdot \alpha| = |\beta \cdot (h \cdot \alpha)|$  If  $\beta = \tau$ :  $|\beta \cdot (h \cdot \alpha)| =$  $|h \cdot \alpha| = |h| \cdot \alpha = |\beta \cdot h| \cdot \alpha$ . If  $\beta \neq \tau$ :  $|\beta \cdot (h \cdot \alpha)| =$  $\beta \cdot |h \cdot \alpha| = \beta \cdot (|h| \cdot \alpha) = (\beta \cdot |h|) \cdot \alpha = |\beta \cdot h| \cdot \alpha.$ 

### **B.10** Preservation of *last*

*Proof.* By structural induction on s. Let  $s, s_1, \rho, \rho_1, h, h_1, \alpha$  be given.

- **case**  $s_1$ ;  $s_2$ : There are two cases for the transition from  $\langle s_1; s_2, \rho, h \rangle$ : ISEQ1 and ISEQ2 hence we argue for both:
  - subcase ISEQ1: In this case  $\langle s_1; s_2, \rho, h \rangle \xrightarrow{\alpha} \langle s_3; s_2, \rho_1, h_1 \rangle$ and  $\langle s_1, \rho, h \rangle \xrightarrow{\alpha} \langle s_3, \rho_1, h_1 \rangle$  and clearly  $last(s_2) =$  $last(s_3; s_2).$
  - subcase ISEQ2: In this case  $\langle s_1; s_2, \rho, h \rangle \xrightarrow{\alpha} \langle s_2, \rho_1, h_1 \rangle$ and clearly  $last(s_2) = last(s_1; s_2)$ .
- case if  $b^{\ell}$  then  $s_1$  else  $s_2$ : There are two cases for the transition from (if  $b^{\ell}$  then  $s_1$  else  $s_2, \rho, h$ ): IIF1 and IIF2 hence we argue for both:
  - subcase IIF1: In this case (if  $b^{\ell}$  then  $s_1$  else  $s_2, \rho, h$ )  $\xrightarrow{\tau}$  $\langle s_1, \rho, h \cdot \tau \rangle$  and  $\rho \vdash_{\mathcal{B}} b \Downarrow$  tt and clearly  $last(s_1) \subseteq$  $last(s_1) \cup last(s_2) = last(if b^{\ell} then s_1 else s_2).$ subcase IIF2: Analogous to subcase IIF1 above.
- **case** while  $b^{\ell}$  do  $s_1$  end: In this case the transition must be by rule IWHILE1 as IWHILE2 leads to a terminal configuration. Hence  $\langle \text{while } b^{\ell} \text{ do } s_1 \text{ end}, \rho, h \rangle \xrightarrow{\tau} \langle s_1; \text{while } b^{\ell} \text{ do } s_1 \text{ end}, \rho, h \cdot \tau \rangle \langle \rho[x \mapsto v], h \cdot ch?v \rangle$  by rule IREAD. Furthermore  $h_1 = h \cdot ch?v \rangle$ and  $\rho \vdash_{\mathcal{B}} b \Downarrow$  tt and  $last(s_1; \texttt{while } b^\ell \texttt{ do } s_1 \texttt{ end}) =$  $last(while b^{\ell} \text{ do } s_1 \text{ end}) = \{\ell\}.$
- **case**  $s_1 \oplus^{\ell} s_2$ : There are two cases for the transition from  $\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle$ : ICHOICE1 and ICHOICE2 hence we argue for both:
  - subcase ICHOICE1: In this case  $\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle \xrightarrow{\alpha} ic_1$ and  $\langle s_1, \rho, h \rangle \xrightarrow{\alpha} ic_1$  for some non-terminal configuration  $ic_1 = \langle s_3, \rho_1, h_1 \rangle$ . By the induction hypothesis  $last(s_3) \subseteq$  $last(s_1) \subseteq last(s_1) \cup last(s_2) = last(s_1 \oplus^{\ell} s_2).$
  - subcase ICHOICE2: Symmetric to subcase ICHOICE1 above.
- other cases: The cases for  $skip^{\ell}$ ,  $x := {}^{\ell}e$ ,  $ch?^{\ell}x$ , and  $ch!^{\ell}e$  are vacuously true as they can only transition to a terminal configuration.

### **B.11** Local soundness of store and history specification 1

*Proof.* By structural induction on s. Let s,  $\rho$ ,  $\rho_1$ , h,  $h_1$ ,  $\alpha$ ,  $\hat{\mathcal{E}}$ ,  $\hat{\mathcal{X}}$ be given. Let  $\ell = first(s)$  and assume  $\langle s, \rho, h \rangle \xrightarrow{\alpha} \langle \rho_1, h_1 \rangle$ ,  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s$ , and that  $\rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)), |h| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell)), \text{ and } |\overline{\alpha} \cdot f| \in \mathcal{I}(\widehat{\mathcal{E}}_{h}(\ell))$  $\mathcal{L}(\mathcal{E}_{f}(\ell)).$ 

- case  $\operatorname{skip}^{\ell}$ :  $(\operatorname{skip}^{\ell}, \rho, h) \xrightarrow{\tau} \langle \rho, h \cdot \tau \rangle, h_1 = h \cdot \tau, \operatorname{first}(\operatorname{skip}^{\ell}) =$  $\ell$ ,  $last(skip^{\ell}) = \{\ell\}$ , and by assumption  $\rho_1 = \rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)) \subseteq$  $\gamma_{st}(\widehat{\mathcal{X}_{\rho}}(\ell)), |h \cdot \tau| = |h| \in \mathcal{L}(\widehat{\mathcal{E}_{h}}(\ell)) \subseteq \mathcal{L}(\widehat{\mathcal{X}_{h}}(\ell)), \text{ and }$  $|\overline{\tau} \cdot f| = |\tau \cdot f| = |f| \in \mathcal{L}(\widehat{\mathcal{L}}_{f}(\ell)) \subseteq \mathcal{L}(\widehat{\mathcal{X}}_{f}(\ell)).$
- case  $x := {}^{\ell}e: \langle x := {}^{\ell}e, \rho, h \rangle \xrightarrow{\tau} \langle \rho[x \mapsto v], h \cdot \tau \rangle$  for some vsuch that  $\rho \vdash_{\mathcal{A}} e \Downarrow v$ . Again  $h_1 = h \cdot \tau$ , first $(x :=^{\ell} e) = \ell$ , and  $last(skip^{\ell}) = \{\ell\}$ . Let  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$ . By assumption and Lemma 7.2 we have  $v \in \gamma_v(\widehat{\mathcal{A}}(\widehat{\rho}, e))$ . But then by assumption and Lemma 7.3  $ho_1 = 
  ho[x \mapsto v] \in$  $\gamma_{st}(\widehat{assign}(\widehat{\rho}, x, \widehat{\mathcal{A}}(\widehat{\rho}, e))) \subseteq \gamma_{st}(\widehat{\mathcal{X}_{\rho}}(\ell)), \ |h \cdot \tau| = |h| \in$  $\mathcal{L}(\widehat{h}) \subseteq \mathcal{L}(\widehat{\mathcal{X}_h}(\ell)), \text{ and } |\overline{\tau} \cdot f| = |\tau \cdot f| = |f| \in \mathcal{L}(\widehat{f}) \subseteq$  $\mathcal{L}(\widehat{\mathcal{X}_f}(\ell)).$

- case while  $b^{\ell}$  do  $s_1$  end: The first premise must be by application of IWHILE2, as IWHILE1 leads to a non-terminal configuration, hence (while  $b^{\ell}$  do  $s_1$  end,  $\rho, h$ )  $\xrightarrow{\tau}$  ( $\rho, h \cdot \tau$ ) and  $\rho \vdash_{\mathcal{B}} b \Downarrow \texttt{ff. Again } h_1 = h \cdot \tau, first(\texttt{while } b^\ell \texttt{ do } s_1 \texttt{ end}) =$  $\ell$ , and  $last(skip^{\ell}) = \{\ell\}$ . Let  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$ . By assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash$  while  $b^{\ell}$  do  $s_1$  end, we have that  $(\widehat{false}(b,\widehat{\rho}),\widehat{h},\widehat{f}) \subseteq \widehat{\mathcal{X}}(\ell)$ . Hence  $\rho \in false(b,\gamma_{st}(\widehat{\rho})) \subseteq$  $\gamma_{st}(\widehat{false}(b,\widehat{\rho})) \subseteq \widehat{\mathcal{X}}_{\rho}(\ell), |h \cdot \tau| = |h| \in \mathcal{L}(\widehat{h}) \subseteq \mathcal{L}(\widehat{\mathcal{X}}_{h}(\ell)),$ and  $|\overline{\tau} \cdot f| = |\tau \cdot f| = |f| \in \mathcal{L}(\widehat{f}) \subseteq \mathcal{L}(\widehat{\mathcal{X}}_{f}(\ell)).$
- **case**  $s_1 \oplus^{\ell} s_2$ : In this case one of two rules: ICHOICE1 or ICHOICE2 was applied:
  - subcase ICHOICE1:  $\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle \xrightarrow{\alpha} ic_1 \text{ and } \langle s_1, \rho, h \rangle \xrightarrow{\alpha}$  $ic_1$  for some terminal configuration  $ic_1 = \langle \rho_1, h_1 \rangle$ . Furthermore  $\ell = first(s_1 \oplus^{\ell} s_2)$  and let  $\ell_1 \in last(s_1) \subseteq$  $last(s_1 \oplus^{\ell} s_2)$ . By assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash s_1 \oplus^{\ell} s_2$  and therefore  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}(\ell) \sqsubseteq \widehat{\mathcal{E}}(first(s_1))$ . Hence by the induction hypothesis  $h_1 = h \cdot \alpha$ , and  $\rho_1 \in \gamma_{st}(\widehat{\mathcal{X}}_{\rho}(\ell_1))$ ,  $|h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{X}_h}(\ell_1)) |s| \in \mathcal{L}(\widehat{\mathcal{X}_f}(\ell_1)).$  When combined with  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{X}}(\ell)$  this yields the desired result.
  - subcase ICHOICE2: The argument is symmetric to the ICHOICE1 case.
- case  $ch?^{\ell}x$ : We first explore the consequences of our assumptions, and then finally conclude each of the three parts of the lemma's right-hand-side on this basis.  $\langle ch?^{\ell}x, \rho, h \rangle \xrightarrow{ch?v}$ ch?v,  $first(ch?^{\ell}x) = \ell$ , and  $last(ch?^{\ell}x) = \{\ell\}$ . Let  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \{\ell\}$ .  $\widehat{\mathcal{E}}(\ell)$ . By assumption  $|\overline{ch?v} \cdot f| = |ch!v \cdot f| = ch!v \cdot |f| \in$  $\mathcal{L}(\widehat{f})$  hence  $|f| \in \mathcal{L}(\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch!v\})}(\widehat{f}))$  by Lemma 4.1. Furthermore [ch; ch] is an atom in *Interval*,  $\alpha_v(\{v\})$  is an atom in Val,  $([ch; ch], \alpha_v(\{v\}))$  is an atom in Interval \* Val, and

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

is an atom in Ch(Val). Hence there exists an equivalence class  $[ch!\hat{v}_a] \in \widehat{range}(\widehat{f})$  such that  $\alpha_{ch}(\{ch!v\}) = ch!(\alpha_v(\{v\})) \in$  $[ch!\widehat{v}_a]$  and  $\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch!v\})}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_a])}(\widehat{f})$ . Therefore we must have  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\hat{v}_a])}(\widehat{f}) \not\subseteq \emptyset$ . Furthermore, by our assumption about project:  $\alpha_{ch}(\{ch!v\}) = ch!(\alpha_v(\{v\})) \sqsubseteq$  $\widetilde{project}([ch!\widehat{v}_a]) = ch!\widehat{v}$  and therefore

$$ch! v \in \gamma_{ch}(project([ch!\hat{v}_{a}]))$$
  
= $\gamma_{ch}(ch!\hat{v})$   
= $\gamma_{wr}((ch,\hat{v}))$   
={[ch; ch]!v | ch  $\in \gamma_{Int}([ch; ch]) \land v \in \gamma_{v}(\hat{v})$ }

which means that  $v \in \gamma_v(\hat{v})$ . But then by the implication in the analysis specification  $\widehat{assign}(\widehat{\rho}, x, \widehat{v}) \sqsubseteq \mathcal{X}_{\rho}(\ell), \widehat{h} \cdot ch? \widehat{v} \sqsubseteq$  $\widehat{\mathcal{X}}_{h}(\ell)$ , and  $\widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_{a}])}(\widehat{f}) \sqsubseteq \widehat{\mathcal{X}}_{f}(\ell)$ .

(Part 1):  $h_1 = h \cdot ch?v$  follows immediately from the above. (Part 2): By assumption  $\rho \in \gamma_{st}(\hat{\rho})$  and since  $v \in \gamma_v(\hat{v})$ Lemma 7.3 yields  $\rho[x \mapsto v] \in \gamma_{st}(assign(\hat{\rho}, x, \hat{v})) \sqsubseteq \mathcal{X}_{\rho}(\ell)$ . (Part 3): By assumption  $|h| \in \mathcal{L}(\widehat{h})$  and since  $v \in \gamma_v(\widehat{v})$ we have  $ch?v \in \gamma_{rd}([ch; ch], \hat{v})$  and furthermore  $ch?v \in$  $\gamma_{ch}(ch?\hat{v})$  and therefore by Lemma 7.7  $|h \cdot ch?v| = |h| \cdot$   $\begin{array}{ll} ch?v \in \mathcal{L}(\widehat{h}) \cdot \{ch?v\} \subseteq \mathcal{L}(\widehat{h}) \cdot \gamma_{ch}(ch?\widehat{v}) = \mathcal{L}(\widehat{h}) \cdot \\ \mathcal{L}(ch?\widehat{v}) = \mathcal{L}(\widehat{h} \cdot ch?\widehat{v}) \subseteq \widehat{\mathcal{X}_{h}}(\ell). \\ (\text{Part 4}): \text{But then } |f| \in \mathcal{L}(\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch!v\})}(\widehat{f})) = \mathcal{L}(\widehat{\mathcal{D}}_{\widehat{repr}([ch!\widehat{v}_{a}])}(\widehat{f})) \\ \subseteq \mathcal{L}(\widehat{\mathcal{X}_{f}}(\ell)). \end{array}$ 

**case**  $ch!^{\ell}e$ : Again we first explore the consequences of our assumptions.  $\langle ch!^{\ell}e, \rho, h \rangle \xrightarrow{ch!v} \langle \rho, h \cdot ch!v \rangle$  where  $\rho \vdash_{\mathcal{A}} e = v$ by rule IWRITE. Furthermore  $h_1 = h \cdot ch!v$ ,  $first(ch!^{\ell}e) = \ell$ , and  $last(ch!^{\ell}e) = \{\ell\}$ . Let  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$ . By assumption  $|\overline{ch!v} \cdot f| = |ch?v \cdot f| = ch?v \cdot |f| \in \mathcal{L}(\widehat{f})$  hence  $|f| \in \mathcal{L}(\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch?v\})}(\widehat{f}))$  by Lemma 4.1. Furthermore

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

is an atom in  $\widehat{Ch}(\widehat{Val})$ . Hence there exists an equivalence class  $[ch?\hat{v}_a] \in \widehat{range}(\widehat{f})$  such that  $\alpha_{ch}(\{ch?v\}) = ch?(\alpha_v(\{v\}\}) \in [ch?\hat{v}_a]$  and  $\widehat{\mathcal{D}}_{\alpha_{ch}(\{ch?v\})}(\widehat{f}) = \widehat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}_a])}(\widehat{f})$ .

Therefore we must have  $\widehat{\mathcal{D}_{repr([ch?\hat{v}_a])}(f)} \not\subseteq \emptyset$ . Furthermore, by our assumption about  $project: \alpha_{ch}(\{ch?v\}) = ch?(\alpha_v(\{v)\}) \sqsubseteq project([ch?\hat{v}_a]) = ch?\hat{v}$  and therefore

$$ch?v \in \gamma_{ch}(\widetilde{project}([ch?\hat{v}_{a}]))$$

$$= \gamma_{ch}(ch?\hat{v})$$

$$= \gamma_{rd}((ch,\hat{v}))$$

$$= \{[ch;ch]?v \mid ch \in \gamma_{Int}([ch;ch]) \land v \in \gamma_{v}(\hat{v})\}$$

which means that  $v \in \gamma_v(\hat{v})$ . But by Lemma 7.2  $v \in \gamma_v(\hat{\mathcal{A}}(\hat{\rho}, e)) = \gamma_v(\hat{v}')$  for  $\hat{\mathcal{A}}(\hat{\rho}, e) = \hat{v}'$  hence  $v \in \gamma_v(\hat{v}) \cap \gamma_v(\hat{v}') = \gamma_v(\hat{v} \sqcap \hat{v}')$  which means that  $\hat{v} \sqcap \hat{v}' \neq \bot$ . But then by the implication in the analysis specification  $\hat{\rho} \sqsubseteq \hat{\mathcal{X}}_{\rho}(\ell)$ ,  $\hat{h} \cdot ch!(\hat{v} \sqcap \hat{v}') \sqsubseteq \hat{\mathcal{X}}_h(\ell)$ , and  $\hat{\mathcal{D}}_{\widehat{repr}([ch?\hat{v}_a])}(\hat{f}) \sqsubseteq \hat{\mathcal{X}}_f(\ell)$ . (Part 1): We immediately have  $h_1 = h \cdot ch!v$  from the above.

(Part 2): We immediately have  $\rho \in \gamma_{st}(\hat{\rho}) \subseteq \widehat{\mathcal{X}}_{\rho}(\ell)$  by assumption.

(Part 3): By assumption  $h \in \mathcal{L}(\widehat{h})$  and since  $v \in \gamma_v(\widehat{v} \sqcap \widehat{v}')$ we have  $ch!v \in \gamma_{wr}([ch; ch], \widehat{v} \sqcap \widehat{v}')$  and furthermore  $ch!v \in \gamma_{ch}(ch!(\widehat{v} \sqcap \widehat{v}'))$  and therefore by Lemma 7.7  $|h \cdot ch!v| = |h| \cdot ch!v \in \mathcal{L}(\widehat{h}) \cdot \{ch!v\} \subseteq \mathcal{L}(\widehat{h}) \cdot \gamma_{ch}(ch!(\widehat{v} \sqcap \widehat{v}')) = \mathcal{L}(\widehat{h}) \cdot \mathcal{L}(ch!(\widehat{v} \sqcap \widehat{v}')) = \mathcal{L}(\widehat{h} \cdot ch!(\widehat{v} \sqcap \widehat{v}')) \subseteq \mathcal{L}(\widehat{\mathcal{X}}_h(\ell))$ (Part 4): By the above we have  $|f| \in \mathcal{L}(\widehat{D}_{\alpha_{ch}(\{ch!v\}}(\widehat{f})) = (f_{ch}) = (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) = (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) = (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) = (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) + (f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) + (f_{ch}) \cdot \mathcal{L}(f_{ch}) + (f_{ch}) + (f_$ 

 $\mathcal{L}(\widehat{\mathcal{D}}_{\widehat{repr}([ch?\widehat{v}_{a}])}(\widehat{f})) \subseteq \mathcal{L}(\widehat{\mathcal{X}}_{f}(\ell)).$ 

case  $\mathtt{stop}^{\ell}$ : Vacuously true as no rules lets us transition from  $\mathtt{stop}^{\ell}$ .

other cases: The remaining cases are vacuously true as input configurations with  $s_1$ ;  $s_2$  and if  $b^{\ell}$  then  $s_1$  else  $s_2$  are related only to non-terminal output configurations.

# 

# B.12 Local soundness of store and history specification 2

*Proof.* By structural induction on s. Let  $s, s_1, \rho, \rho_1, h, h_1, \alpha, \widehat{\mathcal{E}}, \widehat{\mathcal{X}}$ be given and let  $\ell = first(s)$ . Assume  $\langle s, \rho, h \rangle \xrightarrow{\alpha} \langle s_1, \rho_1, h_1 \rangle$ ,  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s, \rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)), |h| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell)), \text{ and } |\overline{\alpha} \cdot f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell)).$  **case**  $s_1$ ;  $s_2$ : The first premise must be by application of either ISEQ1 or ISEQ2 so we argue for both cases:

sι

**Ibcase ISEQ1:** Then 
$$\langle s_1; s_2, \rho, h \rangle \xrightarrow{\alpha} \langle s_3; s_2, \rho_1, h_1 \rangle$$
 and  $\langle s_1, \rho, h \rangle \xrightarrow{\alpha} \langle s_3, \rho_1, h_1 \rangle$ . Furthermore  $\ell = first(s_1; s_2) = first(s_1) = \ell_1$  and from the analysis specification we therefore have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}(\ell) = \widehat{\mathcal{E}}(\ell_1)$  and therefore we can apply the induction hypothesis and conclude  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_3$ ,  $h_1 = h \cdot \alpha, \rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell_3)), |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell_3)), \text{ and } |f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell_3))$  where  $\ell_3 = first(s_3) = first(s_3; s_2)$ .  
(Part 1): We need to show  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_3$ ;  $s_2$  or equivalently (a)  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_3$ , (b)  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_2$ , and (c)  $\widehat{\mathcal{X}}(\ell_3) \sqsubseteq \widehat{\mathcal{E}}(\ell_2)$  for all  $\ell_3' \in last(s_3)$ . (a) follows immediately from the induction hypothesis and (b) follows from the assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1; s_2$ . From the same assumption we furthermore have  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(\ell_2)$  for all  $\ell_1 \in last(s_1)$  which together with  $last(s_3) \subseteq last(s_1)$  from Lemma 7.8 means that  $\widehat{\mathcal{X}}(\ell_3') \sqsubseteq \widehat{\mathcal{E}}(\ell_2)$  for all  $\ell_3' \in last(s_3) \subseteq last(s_1)$  and thus yields (c).  
(Part 2): We have  $h_1 = h \cdot \alpha$  immediately from the above. For parts 3, 4, and 5 since  $\widehat{\mathcal{E}}(\ell_3) = \widehat{\mathcal{E}}(first(s_3)) = \widehat{\mathcal{E}}(first(s_3; s_2)))$ ,  $|h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell_3)) = \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_3; s_2))))$ , and  $|f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell_3)) = \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_3; s_2))))$ .

**subcase ISEQ2:** Then  $\langle s_1; s_2, \rho, h \rangle \xrightarrow{\alpha} \langle s_2, \rho_1, h_1 \rangle$  and  $\langle s_1, \rho, h \rangle \xrightarrow{\alpha} \langle \rho_1, h_1 \rangle$  Furthermore from the analysis specification we have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_2$  and for all  $\ell_1 \in last(s_1). \ \widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(first(s_2))$  by assumption. By Lemma 7.9 we can conclude  $h_1 = h \cdot \alpha, \rho_1 \in \gamma_{st}(\widehat{\mathcal{X}}_{\rho}(\ell_1)), |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{X}}_{h}(\ell_1)), \text{ and } |f| \in \mathcal{L}(\widehat{\mathcal{X}}_{f}(\ell_1)).$  for all  $\ell_1 \in last(s_1).$ 

(Part 1): We immediately have  $\hat{\mathcal{E}}, \hat{\mathcal{X}} \models s_2$  from the analysis specification.

(Part 2): We immediately have  $h_1 = h \cdot \alpha$  from the above. Parts 3, 4, and 5 follow from combining the above facts:  $\rho_1 \in \gamma_{st}(\widehat{\mathcal{X}}_{\rho}(\ell_1)) \subseteq \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_2))), |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{X}}_{h}(\ell_1))$  $\subseteq \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_2))), \text{ and } |f| \in \mathcal{L}(\widehat{\mathcal{X}}_{f}(\ell_1)) \subseteq \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_2))).$ 

**case** if  $b^{\ell}$  then  $s_1^{\ell_1}$  else  $s_2^{\ell_2}$ : The first premise must be by application of either IIF1 or IIF2 so we argue for both cases:

**subcase IIF1:** Then  $\langle \text{if } b^{\ell} \text{ then } s_1 \text{ else } s_2, \rho, h \rangle \xrightarrow{\tau} \langle s_1, \rho, h \cdot \tau \rangle$ and  $\rho \vdash_{\mathcal{B}} b \Downarrow \text{tt}$  and  $first(\text{if } b^{\ell} \text{ then } s_1 \text{ else } s_2) = \ell$ . By assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models \text{if } b^{\ell} \text{ then } s_1 \text{ else } s_2, \rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell)), |h| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(\ell)), \text{ and } |\overline{\alpha} \cdot f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(\ell)).$  Let  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$ . (Part 1): We immediately have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  from the assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models \text{if } b^{\ell} \text{ then } s_1 \text{ else } s_2$ . (Part 2): We immediately have  $h_1 = h \cdot \tau$  from the above. (Part 3): Since  $\rho \in \gamma_{st}(\widehat{\rho})$  and  $\rho \vdash_{\mathcal{B}} b \Downarrow \text{tt}$  we have  $\rho \in \gamma_{st}(\widehat{true}(\widehat{\rho}, b)) \subseteq \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1)))$  by Lemma 7.4 and the analysis specification. (Part 4): By assumption  $|h| \in \mathcal{L}(\widehat{h})$  and Lemma 7.7  $|h_1| =$  $|h \cdot \tau| = |h| \in \mathcal{L}(\widehat{h}) \subseteq \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_1)))$ . (Part 5): By assumption we get  $|\overline{\tau} \cdot f| = |\tau \cdot f| = |f| \in$  $\mathcal{L}(\widehat{f}) \subseteq \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_1)))$ .

- subcase IIF2: The argument is symmetric to that of subcase IIF1.
- **case** while  $b^{\ell}$  do  $s_1$  end: The first premise must be by application of IWHILE1, as IWHILE2 leads to a terminal configuration, hence we have  $\langle \text{while } b^{\ell} \text{ do } s_1 \text{ end}, \rho, h \rangle \xrightarrow{\tau}$

 $\langle s_1 ; \text{while } b^\ell \text{ do } s_1 \text{ end}, \rho, h \cdot \tau \rangle$  and  $\rho \vdash_{\mathcal{B}} b \Downarrow \text{tt}$  and  $first(\text{while } b^\ell \text{ do } s_1 \text{ end}) = \ell$ .

(Part 1): By assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$ ,  $(\widehat{true}(b, \widehat{\rho}), \widehat{h}, \widehat{f}) \sqsubseteq \widehat{\mathcal{E}}(first(s_1))$ , and for all  $\ell_1 \in last(s_1)$ .  $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(\ell)$  where  $(\widehat{\rho}, \widehat{h}, \widehat{f}) = \widehat{\mathcal{E}}(\ell)$ . To satisfy  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$ ; while  $b^{\ell}$  do  $s_1$  end we need  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1, \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models$  while  $b^{\ell}$  do  $s_1$  end, and for all  $\ell_1 \in last(s_1)$   $\widehat{\mathcal{X}}(\ell_1) \sqsubseteq \widehat{\mathcal{E}}(first(while b^{\ell} \text{ do } s_1 \text{ end})) = \widehat{\mathcal{E}}(\ell)$  which all follow by assumption.

(Part 2): We immediately have  $\hat{h}_1 = h \cdot \tau$  from the above.

(Part 3): Since  $\rho \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(\ell))$  and  $\rho \vdash_{\mathcal{B}} b \Downarrow$  the we have  $\rho \in \gamma_{st}(\widehat{true}(\widehat{\mathcal{E}}_{\rho}(\ell), b)) \subseteq \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1)))$  by Lemma 7.4 and the analysis specification.

(Part 4): By assumption  $|h| \in \mathcal{L}(\widehat{\mathcal{E}_h}(\ell))$  and Lemma 7.7  $|h_1| = |h \cdot \tau| = |h| \in \mathcal{L}(\widehat{\mathcal{E}_h}(\ell)) \subseteq \mathcal{L}(\widehat{\mathcal{E}_h}(first(s_1))).$ (Part 5): By assumption we get  $|\overline{\tau} \cdot f| = |\tau \cdot f| = |f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(\ell)) \subseteq \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_1))).$ 

**case**  $s_1 \oplus^{\ell} s_2$ : In this case one of two rules: ICHOICE1 or ICHOICE2 was applied:

**subcase ICHOICE1:**  $\langle s_1 \oplus^{\ell} s_2, \rho, h \rangle \xrightarrow{\alpha} ic_1 \text{ and } \langle s_1, \rho, h \rangle \xrightarrow{\alpha} ic_1 \text{ for some non-terminal configuration } ic_1 = \langle s_3, \rho_1, h_1 \rangle$ and  $first(s_1 \oplus^{\ell} s_2) = \ell$ . By assumption  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1 \oplus^{\ell} s_2$ and therefore  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1$  and  $\widehat{\mathcal{E}}(\ell) \sqsubseteq \widehat{\mathcal{E}}(first(s_1))$ . Hence by the induction hypothesis  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_3, h_1 = h \cdot \alpha,$  $\rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_3))), |h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_3)))$  $|f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_3))).$ 

(Part 1): We immediately have  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_3$  from the induction hypothesis.

(Part 2): We immediately have  $h_1 = h \cdot \tau$  from the above. (Part 3): We immediately have  $\rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_3)))$ 

from the above. (Part 4): We immediately have  $|h \cdot \alpha| \in \mathcal{L}(\widehat{\mathcal{E}}_h(first(s_3)))$  from the above.

(Part 5): We immediately have  $|f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_3)))$  from the above.

**subcase ICHOICE2:** The argument is symmetric to the ICHOICE1 case.

other cases: The remaining cases are vacuously true as input configurations with  $\text{skip}^{\ell}$ ,  $x := {}^{\ell}e$ ,  $ch?^{\ell}x$ ,  $ch!^{\ell}e$ , and  $\text{stop}^{\ell}$  are related only to terminal output configurations.

### B.13 Non-terminating system analysis soundness

Proof. By induction on the length of the decoupled trace.

- **case** n = 1: Then  $ic_1 = \langle s_1, \rho_1, h_1 \rangle = ic_n = \langle s_n, \rho_n, h_n \rangle$ which means that  $s_1 = s_n, \ell_1 = first(s_1) = first(s_n) = \ell_n$ ,  $\rho_1 = \rho_n$ , and  $h_1 = h_n$ . Furthermore from our assumptions we get  $\rho_1 = \rho_n \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_n))) = \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1)))$ ,  $|h_1\alpha_1\alpha_2...\alpha_0| = |h_1\epsilon| = |h_n| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_1))) =$  $\mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_n)))$ , and  $|\beta_1\beta_2...\beta_{n-1}f| = |\beta_1\beta_2...\beta_0f| =$  $|f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_1))) = \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_n)))$ .
- **case** n = k + 1: We proceed by case analysis on  $\alpha_1$ . **subcase**  $\alpha_1 = \epsilon$ : Then  $\beta_1 = \tau$  by rule ISYSRIGHT since otherwise the first step would not represent a decoupled system step. But then  $ic_1 = ic_2 = \langle s_1, \rho_1, h_1 \rangle$ . We assume  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \models s_1, \rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1))), |h_1| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_1))), and |\beta_1\beta_2...\beta_{n-1}f| = |\tau\beta_2...\beta_{n-1}f| = |\beta_2...\beta_{n-1}f| \in \mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_1)))$ . By the induction hypothesis we therefore have  $\rho_n \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_n))), |h_1\epsilon\alpha_2...\alpha_{n-1}| =$

 $|h_1\alpha_1\alpha_2\ldots\alpha_{n-1}| = |h_n| \in \mathcal{L}(\widehat{\mathcal{E}_h}(first(s_n))), \text{ and } |f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_n))).$ 

**subcase**  $\alpha_1 \neq \epsilon$ : It must be that case that  $ic_2 = \langle s_2, \rho_2, h_2 \rangle$ since no rules allow transitions out of terminal configurations  $\langle \rho_2, h_2 \rangle$ . Since  $\langle s_1, \rho_1, h_1 \rangle \xrightarrow{\alpha_1} \langle s_2, \rho_2, h_2 \rangle, \widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash$  $s_1, \rho_1 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_1))), |h_1| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_1))),$ and  $|\beta_1\beta_2...\beta_{n-1}f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_1)))$  we can conclude from Lemma 7.10 that  $\widehat{\mathcal{E}}, \widehat{\mathcal{X}} \vDash s_2, h_2 = h_1 \cdot \alpha_1,$  $\rho_2 \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_2))), |h_1 \cdot \alpha_1| \in \mathcal{L}(\widehat{\mathcal{E}}_{h}(first(s_2))),$ and  $|\beta_2 \dots \beta_{n-1}f| \in \mathcal{L}(\widehat{\mathcal{E}_f}(first(s_2)))$ . In particular, for  $\alpha_1 = \tau$  then  $\beta_1 = \epsilon$  since the first step must be an application of ISYSLEFT. Similarly if  $\alpha_1 \neq \tau$  then  $\beta_1 = \overline{\alpha_1}$ since the first step must be an application of either ISYSWR or ISYSRW. Now the remainder  $\langle s_2, \rho_2, h_2 \rangle = ic_2 \xrightarrow{\alpha_2}$  $\dots \xrightarrow{\alpha_{n-1}} ic_n$  and  $\overline{ic_2} \xrightarrow{\beta_2} \dots \xrightarrow{\beta_{n-1}} \overline{ic_n}$  represents a decoupled system trace. We can therefore apply the induction hypothesis and conclude  $\rho_n \in \gamma_{st}(\widehat{\mathcal{E}}_{\rho}(first(s_n))))$ ,  $|(h_1 \cdot \alpha_1)\alpha_2 \dots \alpha_{n-1}| \in \mathcal{L}(\mathcal{E}_f(first(s_n))), \text{ and } |f| \in$  $\mathcal{L}(\widehat{\mathcal{E}}_{f}(first(s_n)))$  as desired.

# C. Widening proofs

# **C.1** rev twice is identity: $\forall r \in \widehat{R}_A$ . rev(rev(r)) = r

*Proof.* By structural induction over r. Let  $r \in \hat{R}_A$  be given.

 $\begin{aligned} & \operatorname{case} \emptyset: \ rev(rev(\emptyset)) = rev(\emptyset) = \emptyset \\ & \operatorname{case} \epsilon: \ rev(rev(\epsilon)) = rev(\epsilon) = \epsilon \\ & \operatorname{case} \ell: \ rev(rev(\ell)) = rev(\ell) = \ell \\ & \operatorname{case} r_1 \cdot r_2: \ rev(rev(r_1 \cdot r_2)) = rev(rev(r_2) \cdot rev(r_1)) = rev(rev(r_1)) \\ & rev(rev(r_2)) = r_1 \cdot r_2 \text{ by two applications of the IH.} \\ & \operatorname{case} r_1^*: \ rev(rev(r_1^*)) = rev(rev(r_1)^*) = rev(rev(r_1))^* = r_1^* \\ & \operatorname{by applying the IH.} \\ & \operatorname{case} r_1 + r_2: \ rev(rev(r_1 + r_2)) = \ rev(rev(r_1) + rev(r_2)) = \\ & rev(rev(r_1)) + rev(rev(r_2)) = r_1 + r_2 \text{ by two applications of the IH.} \\ & \operatorname{case} r_1 \& r_2: \ rev(rev(r_1 \& r_2)) = \ rev(rev(r_1) \& rev(r_2)) = \\ & rev(rev(r_1)) \& rev(rev(r_2)) = r_1 \& r_2 \text{ by two applications of the IH.} \\ & \operatorname{case} \mathbb{C} r_1: \ rev(rev(\mathbb{C} r_1)) = \ rev(\mathbb{C} rev(r_1)) = \mathbb{C} rev(rev(r_1)) = \\ & \mathbb{C} r_1 \text{ by applying the IH.} \end{aligned}$ 

**C.2** rev is reverse:  $\forall r \in \widehat{R}_A$ .  $\mathcal{L}(rev(r)) = rev(\mathcal{L}(r))$ where  $rev(S) = \{rev(s) \mid s \in S\}$ 

*Proof.* By structural induction over r. Let  $r \in \widehat{R}_A$  be given.

**case**  $\emptyset$ :  $\mathcal{L}(rev(\emptyset)) = \mathcal{L}(\emptyset) = \emptyset = rev(\emptyset) = rev(\mathcal{L}(\emptyset))$  by the definitions of rev and  $\mathcal{L}$ . **case**  $\epsilon$ :  $\mathcal{L}(rev(\epsilon)) = \mathcal{L}(\epsilon) = \{\epsilon\} = rev(\{\epsilon\}) = rev(\mathcal{L}(\epsilon))$  by the definitions of rev and  $\mathcal{L}$ . **case**  $\ell$ :  $\mathcal{L}(rev(\ell)) = \mathcal{L}(\ell) = \{c \mid c \in \gamma(\ell)\} = rev(\{c \mid c \in \gamma(\ell)\}) = rev(\mathcal{L}(\ell))$  by the definitions of rev and  $\mathcal{L}$ .

case  $r_1 \cdot r_2$ :

$\mathcal{L}(rev(r_1 \cdot r_2))$	
$= \mathcal{L}(rev(r_2) \cdot rev(r_1))$	(by def. of rev)
$= \mathcal{L}(rev(r_2)) \cdot \mathcal{L}(rev(r_1))$	(by def. of $\mathcal{L}$ )
$= rev(\mathcal{L}(r_2)) \cdot rev(\mathcal{L}(r_1))$	(by IH, twice)
$= rev(\mathcal{L}(r_1) \cdot \mathcal{L}(r_2))$	(by def. of rev)
$= rev(\mathcal{L}(r_1 \cdot r_2))$	(by def. of $\mathcal{L}$ )

case  $r_1^*$ :

$$\mathcal{L}(rev(r_1^*)) = \mathcal{L}(rev(r_1)^*) \qquad (by \text{ def. of } rev) = \bigcup_{i \ge 0} \mathcal{L}(rev(r_1))^i \qquad (by \text{ def. of } \mathcal{L}) = \bigcup_{i \ge 0} rev(\mathcal{L}(r_1))^i \qquad (by \text{ IH}) = rev(\bigcup_{i \ge 0} \mathcal{L}(r_1)^i) \qquad (by \text{ def. of } rev) = rev(\mathcal{L}(r_1^*)) \qquad (by \text{ def. of } \mathcal{L})$$

### case $r_1 + r_2$ :

$$\begin{aligned} \mathcal{L}(rev(r_1 + r_2)) \\ &= \mathcal{L}(rev(r_1) + rev(r_2)) & \text{(by def. of } rev) \\ &= \mathcal{L}(rev(r_1)) \cup \mathcal{L}(rev(r_2)) & \text{(by def. of } \mathcal{L}) \\ &= rev(\mathcal{L}(r_1)) \cup rev(\mathcal{L}(r_2)) & \text{(by IH, twice)} \\ &= rev(\mathcal{L}(r_1) \cup \mathcal{L}(r_2)) & \text{(by def. of } rev) \\ &= rev(\mathcal{L}(r_1 + r_2)) & \text{(by def. of } \mathcal{L}) \end{aligned}$$

case  $r_1 \& r_2$ :

$$\mathcal{L}(rev(r_1 \& r_2))$$

$$= \mathcal{L}(rev(r_1) \& rev(r_2)) \qquad (by \text{ def. of } rev)$$

$$= \mathcal{L}(rev(r_1)) \cap \mathcal{L}(rev(r_2)) \qquad (by \text{ def. of } \mathcal{L})$$

$$= rev(\mathcal{L}(r_1)) \cap rev(\mathcal{L}(r_2)) \qquad (by \text{ IH, twice})$$

$$= rev(\mathcal{L}(r_1) \cap \mathcal{L}(r_2)) \qquad (by \text{ def. of } rev)$$

$$) \cdot \qquad = rev(\mathcal{L}(r_1 \& r_2)) \qquad (by \text{ def. of } \mathcal{L})$$

case  $Cr_1$ :

**C.3** rev is monotone:  $\forall r, r' \in \widehat{R}_A. r \sqsubseteq r' \implies rev(r) \sqsubseteq rev(r')$  *Proof.* We prove  $\mathcal{L}(r) \subseteq \mathcal{L}(r') \implies \mathcal{L}(rev(r)) \subseteq \mathcal{L}(rev(r)).$ Let  $r, r' \in \widehat{R}_A$  be given and assume  $\mathcal{L}(r) \subseteq \mathcal{L}(r')$ .

$$\mathcal{L}(rev(r)) = rev(\mathcal{L}(r)) \qquad (by \text{ Lemma 5.2})$$

$$\subseteq rev(\mathcal{L}(r')) \qquad (by \text{ monotonicity of } rev : \wp(C^*) \longrightarrow \wp(C^*))$$

$$= \mathcal{L}(rev(r')) \qquad (by \text{ Lemma 5.2})$$

# C.4 Reversing a widening operator over regular expressions *Proof.*

 $\nabla_{rev}$  is increasing in both arguments:

By assumption  $\forall r, r' \in \widehat{R}_A$ .  $r \sqsubset r \bigtriangledown r'$ . But then  $rev(r) \sqsubset rev(r) \bigtriangledown rev(r')$  and hence by monotonicity of  $rev: r = rev(rev(r)) \sqsubset rev(rev(r) \lor rev(r')) = r \bigtriangledown r_{rev} r'$ . Similarly, by assumption  $\forall r, r' \in \widehat{R}_A$ .  $r' \sqsubset r \bigtriangledown r'$ . But then  $rev(r') \sqsubset rev(r) \lor rev(r')$  and hence by monotonicity of rev:  $r' = rev(rev(r)) \sqsubset rev(rv) \lor rev(r') = r \bigtriangledown r_{rev} r'$ .

**Chain property:** By assumption, for all increasing chains  $r_0 \sqsubset r_1 \sqsubset r_2 \sqsubset \dots$  the alternative chain defined as  $\overline{r}_0 = r_0$  and  $\overline{r}_{k+1} = \overline{r}_k \lor r_k$  is guaranteed to stabilize after a finite number of steps.

Let an increasing chain  $r_0 \sqsubset r_1 \sqsubset r_2 \sqsubset \ldots$  be given. By monotonicity of *rev*, the pointwise reversed chain is still increasing:  $rev(r_0) \sqsubset rev(r_1) \sqsubset rev(r_2) \sqsubset \ldots$ . But then the alternative chain defined as  $\overline{r}'_0 = rev(r_0)$  and  $\overline{r}'_{k+1} =$  $\overline{r}'_k \bigtriangledown rev(r_k)$  is increasing and guaranteed to stabilize after a finite number of steps. By induction it now follows that  $rev(\overline{r}'_i) = \overline{r}_i$ : **case** 0:  $rev(\overline{r}'_0) = rev(rev(r_0)) = r_0$ .

case $k + 1$ :	
$rev(\overline{r}'_{k+1})$	
$= rev(\overline{r}'_k \triangledown rev(r_k))$	(by def. of $\overline{r}'_{k+1}$ )
$= rev(rev(rev(\overline{r}'_k)) \lor rev(r_k))$	(By Lemma 5.1)
$= rev(\overline{r}'_k)  \triangledown_{rev}  r_k$	(by def. of $\nabla_{rev}$ )
$=\overline{r}_k \bigtriangledown_{rev} r_k$	(by the IH)
$=\overline{r}_{k+1}$	(by def. of $\overline{r}_{k+1}$ )

Hence by monotonicity of *rev*,  $rev(\overline{r}'_0) \sqsubset rev(\overline{r}'_1) \sqsubset \ldots \sqsubset rev(\overline{r}'_{n-1}) \sqsubset rev(\overline{r}'_n)$  for some *n*.