# The Calculational Design of a Generic Abstract Interpreter

Patrick COUSOT
*LIENS, Département de Mathématiques et Informatique*
*École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05, France*

**Abstract.** We present in extenso the calculation-based development of a generic compositional reachability static analyzer for a simple imperative programming language by abstract interpretation of its formal rule-based/structured small-step operational semantics.

## Contents

## 1. Introduction

The 1998 Marktoberdorf international summer school on *Calculational System Design* has been "focusing on techniques and the scientific basis for calculation-based development of software and hardware systems as a foundation for advanced methods and tools for software and system engineering. This includes topics of specification, description, methodology, refinement, verification, and implementation.". Accordingly, the goal of our course was to explain both

- the *calculation-based development of an abstract interpreter* for the automatic static analysis of a simple imperative language, and
- the principles of application of abstract interpretation to the partial verification of programs by *abstract checking*.

For short in these course notes, we concentrate only on the calculational design of a simplified but compositional version of the static analyzer. Despite the fact that the considered imperative language is quite simple and the corresponding analysis problem is supposed to be classical and satisfactorily solved for a long time [9], the proposed analyzer is both compositional and much more precise (e.g. for boolean expressions) than the solutions, often naïve, proposed in the literature. Consequently the results presented is these notes, although quite elementary, go much beyond a mere introductory survey and are of universal use.

A static analyzer takes as input a program written in a given programming language (or a family thereof) and statically, automatically and in finite time[1] outputs an approximate description of all its possible runtime behaviors considered in all possible execution environments (e.g. for all possible input data). The approximation is sound or conservative in that not a single case is forgotten but it may not be precise since the problem of determining the strongest program properties (including e.g. termination) is undecidable.

This automatically determined information can then be compared to a specification either for program transformation, validation, or for error detection [2]. This comparison can also be inconclusive when the automatic analysis is too imprecise. The specification can be provided by the formal semantics of the language which formally defines which errors are detected at runtime or can be defined by the programmer for interactive abstract checking. The purpose of the static analysis is to detect the presence or absence of runtime errors at compile-time, without executing the program. Because the abstract checking is exhaustive, it can detect rare faults which are difficult to come upon by hand. Because the static determination of non-trivial dynamic properties is undecidable, the analysis may also be inconclusive for some tests. By experience, this represents usually from 5 to 20% of the cases which shows that static analysis can considerably reduces the validation task (whether it is done by hand or semi-automatically). See [27] for a recent and successful experience for industrial critical code.

The main idea of abstract interpretation [5, 9, 13] is that any question about a program can be answered using some approximation of its semantics. This approximation idea applies to the semantics themselves [6] which describe program execution at an abstraction level which is often very far from the hardware level but is nevertheless precise enough to conclude e.g. on termination (but not e.g. on exact execution times). The specification of a correct static analyzer and its proof can be understood as an approximation of a semantics, a process which is formalized by the abstract interpretation theory. In the context of the Marktoberdorf summer

[1] from a few seconds for small programs to a few hours for very large programs;

[2] As shown in the course, the situation is not so simple since the analysis and verification do interact.

school, these course notes put the emphasis on viewing abstract interpretation as a formal method for the calculational design of static analyzers for programming languages equipped with a formally defined semantics.

## 2. Definitions

A *poset* $\langle L, \sqsubseteq\rangle$ is a set $L$ with a partial order $\sqsubseteq$ (that is a reflexive, antisymmetric and transitive binary relation on $L$) [20]. A *directed complete partial order* (dcpo) $\langle L, \sqsubseteq, \sqcup\rangle$ is a poset $\langle L, \sqsubseteq\rangle$ such that increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ of elements of $L$ have a least upper bound (lub, join) $\bigsqcup_{i\geq 0} x_i$. A *complete partial order* (cpo) $\langle L, \bot, \sqsubseteq, \sqcup\rangle$ is a dcpo $\langle L, \sqsubseteq, \sqcup\rangle$ with an infimum $\bot = \sqcup\emptyset$. A *complete lattice* $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ is a poset $\langle L, \sqsubseteq\rangle$ such that any subset $X \subseteq L$ has a lub $\sqcup X$. It follows that $\bot = \sqcup\emptyset$ is the infimum, $\top = \sqcup L$ is the supremum and any subset has a greatest lower bound (glb, meet) $\sqcap X = \sqcup\{x \in L \mid \forall y \in X : x \sqsubseteq y\}$.

A map $\mathcal{F} \in L \mapsto L$ of $L$ into $L$ is *monotonic* (written $\mathcal{F} \in L \stackrel{\text{mon}}{\longmapsto} L$) if and only if

$$\forall x, y \in L : x \sqsubseteq y \implies \mathcal{F}(x) \sqsubseteq \mathcal{F}(y) .$$

If $\mathcal{F} \in L \stackrel{\text{mon}}{\longmapsto} L$ is a monotonic map of $L$ into $L$ and $m \sqsubseteq \mathcal{F}(m)$ then $\mathrm{lfp}^{\sqsubseteq}_m \mathcal{F}$ denotes the $\sqsubseteq$*-least fixpoint of $\mathcal{F}$ which is $\sqsubseteq$-greater than or equal to $m$* (if it exists). It is characterized by

$$\begin{array}{rcl}
\mathcal{F}(\mathrm{lfp}^{\sqsubseteq}_m \mathcal{F}) & = & \mathrm{lfp}^{\sqsubseteq}_m \mathcal{F}, \\
m & \sqsubseteq & \mathrm{lfp}^{\sqsubseteq}_m \mathcal{F}, \\
(m \sqsubseteq x) \wedge (\mathcal{F}(x) = x) & \Longrightarrow & \mathrm{lfp}^{\sqsubseteq}_m \mathcal{F} \sqsubseteq x .
\end{array}$$

$\mathrm{lfp}^{\sqsubseteq} \mathcal{F} \stackrel{\Delta}{=} \mathrm{lfp}^{\sqsubseteq}_{\bot} \mathcal{F}$ is the least fixpoint of $\mathcal{F}$. The greatest fixpoint (gfp) is defined dually, replacing $\sqsubseteq$ by its inverse $\sqsupseteq$, the infimum $\bot$ by the supremum $\top$, the lub $\sqcup$ by the greatest lower bound (glb) $\sqcap$, etc.

In order to generalize the Kleene/Knaster/Tarski fixpoint theorem, the transfinite iteration sequence is defined as ($\mathbb{O}$ is the class of ordinals)

$$\begin{array}{rcll}
\mathcal{F}^0(m) & \stackrel{\Delta}{=} & m, & \\
\mathcal{F}^{\delta+1}(m) & \stackrel{\Delta}{=} & \mathcal{F}(\mathcal{F}^{\delta}(m)) & \text{for successor ordinals,} \\
\mathcal{F}^{\lambda}(m) & \stackrel{\Delta}{=} & \bigsqcup_{\delta<\lambda} \mathcal{F}^{\delta}(m) & \text{for limit ordinals .}
\end{array} \tag{1}$$

This increasing sequence $\mathcal{F}^{\delta}$, $\delta \in \mathbb{O}$ is ultimately stationary at rank $\epsilon \in \mathbb{O}$ and converges to $\mathcal{F}^{\epsilon} = \mathrm{lfp}^{\sqsubseteq}_m \mathcal{F}$. This directly leads to an iterative algorithm which is finitely convergent when $L$ satisfies the ascending chain condition (ACC)[3].

The *complement* $\neg P$ of a subset $P \subseteq S$ of a set $S$ is $\{s \in S \mid s \notin P\}$. The *left-restriction* $P \rceil t$ of a relation $t$ on $S$ to $P \subseteq S$ is $\{\langle s, s'\rangle \in t \mid s \in P\}$. The *composition* of relations is $t \circ r \stackrel{\Delta}{=} \{\langle s, s''\rangle \mid \exists s' \in S : \langle s, s'\rangle \in t \wedge \langle s', s''\rangle \in r\}$. The *iterates* of the relation $t$ are defined inductively by

[3] $L$ satisfies the ACC if and only if any strictly ascending chain $x_0 \sqsubset x_1 \sqsubset \cdots$ of elements of $L$ is necessarily finite.

$$\begin{array}{rrcllll} & t^n & \stackrel{\Delta}{=} & \emptyset & & & \text{for } n < 0, \\ & t^0 & \stackrel{\Delta}{=} & 1_S & \stackrel{\Delta}{=} & \{\langle s,\ s\rangle \mid s \in S\} & \text{(that is identity on the set } S\text{)}, \\ \text{and} & t^{n+1} & \stackrel{\Delta}{=} & t \circ t^n & = & t^n \circ t, & \text{for } n \geq 0. \end{array}$$

The *reflexive transitive closure* $t^\star$ of the relation $t$ is

$$t^\star \ \stackrel{\Delta}{=}\ \bigcup_{n\geq 0} t^n \ =\ \bigcup_{n\geq 0}\left(\bigcup_{i\leq n} t^i\right) \ =\ \bigcup_{n\geq 0} (\lambda r\bullet 1_S \cup t \circ r)^i\,(\emptyset) \ =\ \mathrm{lfp}^{\subseteq}\,\lambda r\bullet 1_S \cup t \circ r\ .$$

## 3. Values

### 3.1 Machine integers

We consider a simple but realistic programming language such that the basic values are bounded machine integers. They should satisfy

$$\begin{array}{lll} \texttt{max\_int} > 9, & & \text{greatest machine integer;} \\ \texttt{min\_int} \ \stackrel{\Delta}{=}\ -\texttt{max\_int} - 1, & & \text{smallest machine integer;} \\ z \in \mathbb{Z}, & & \text{mathematical integers;} \\ i \in \mathbb{I} \ \stackrel{\Delta}{=}\ [\texttt{min\_int}, \texttt{max\_int}], & & \text{bounded machine integers.} \end{array} \tag{2}$$

### 3.2 Errors

We assume that the programming language semantics keeps track of uninitialized variables (e.g. by means of a reserved value) and of arithmetic errors (overflow, division by zero, . . . , e.g. by means of exceptions). We use the following notations

$$\begin{array}{lll} \Omega_{\texttt{i}}, & & \text{initialization error;} \\ \Omega_{\texttt{a}}, & & \text{arithmetic error;} \\ e \in \mathbb{E} & \stackrel{\Delta}{=}\ \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}, & \text{errors;} \\ v \in \mathbb{I}_\Omega & \stackrel{\Delta}{=}\ \mathbb{I} \cup \mathbb{E}, & \text{machine values.} \end{array} \tag{3}$$

## 4. Properties of Values

A value property is understood as the set of values which have this property. The concrete properties of values are therefore elements of the powerset $\wp(\mathbb{I}_\Omega)$. For example $[1, \texttt{max\_int}] \in \wp(\mathbb{I}_\Omega)$ is the property "is a positive machine integer" while $\{2n + 1 \in \mathbb{I} \mid n \in \mathbb{Z}\}$ is the property "is an odd machine integer". $\langle \wp(\mathbb{I}_\Omega),\ \subseteq,\ \emptyset,\ \mathbb{I}_\Omega,\ \cup,\ \cap,\ \neg\rangle$ is a complete boolean lattice. Elements of the powerset $\wp(\mathbb{I}_\Omega)$ are understood as predicates or properties of values with subset inclusion $\subseteq$ as logical implication, $\emptyset$ is false, $\mathbb{I}_\Omega$ is true, $\cup$ is the disjunction, $\cap$ is the conjunction and $\neg$ is the negation.

## 5. Abstract Properties of Values

### 5.1 Galois connection based abstraction

For program analysis, we can only use a machine encoding $L$ of a subset of all possible value properties. $L$ is the set of abstract properties. Any abstract property $p \in L$ is the machine encoding of some value property $\gamma(p) \in \wp(\mathbb{I}_\Omega)$ specified by the concretization function $\gamma \in L \mapsto \wp(\mathbb{I}_\Omega)$.

For any particular program to be analyzed, this set can be chosen as a finite set (since there always exists a complete abstraction into a finite abstract domain to prove a specific property of a specific system/program, as shown by the completeness proof given in [16]). However, when considering all programs of a programming language this set $L$ must be infinite (as shown by the incompleteness argument of [16]). This does not mean that $L$ and its meaning $\gamma$ must be the same for all programs in the language (see Sec. 13.4 for a counter-example). But then $L[\![P]\!]$and $\gamma[\![P]\!]$ must be defined for all programs $P$ in the language, not only for a few given ones. This is a fundamental difference with *abstract model checking* where a user-defined problem specific abstraction is considered for each particular system (program) to analyze.

We assume that $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ is a complete lattice so that the partial ordering $\sqsubseteq$ also called approximation ordering is understood as abstract logical implication, the infimum $\bot$ encodes false, the supremum $\top$ encodes true, the lub $\sqcup$ is the abstract disjunction and the glb $\sqcap$ is the abstract conjunction. The fact that the approximation ordering $\sqsubseteq$ should encode logical implication on abstract properties is formalized by the assumption that the concretization function is monotone, that is, by definition

$$p \sqsubseteq q \quad\Longrightarrow\quad \gamma(p) \subseteq \gamma(q)\ . \tag{4}$$

In general, an arbitrary concrete value property $P \in \wp(\mathbb{I}_\Omega)$ has no abstract equivalent in $L$. However it can be overapproximated by any $p \in L$ such that $P \subseteq \gamma(p)$. Overapproximation means that the abstract property $p$ (or its meaning $\gamma(p)$) is weaker than the overapproximated concrete property $P$.

Observe that $\cap\{\gamma(p) \mid P \subseteq \gamma(p)\}$ is a better overapproximation of the concrete property $P$ than any other $p \in L$ such that $P \subseteq \gamma(p)$. The situation where for all concrete properties $P \in \wp(\mathbb{I}_\Omega)$ this best approximation $\cap\{\gamma(p) \mid P \subseteq \gamma(p)\}$ has a corresponding encoding in the abstract domain $L$ corresponds to Galois connections [13]. This encoding of the best approximation is provided by the abstraction function $\alpha \in \wp(\mathbb{I}_\Omega) \mapsto L$ such that

$$\begin{aligned}
&P \subseteq Q \quad\Longrightarrow\quad \alpha(P) \sqsubseteq \alpha(Q) && (\alpha \text{ preserves implication}), && (5)\\
&\forall P \in \wp(\mathbb{I}_\Omega) : P \subseteq \gamma(\alpha(P)) && (\alpha(P) \text{ overapproximates } P), && (6)\\
&\forall p \in \alpha(\gamma(p)) \sqsubseteq p && (\gamma \text{ introduces no loss of information}). && (7)
\end{aligned}$$

Observe that if $p \in L$ overapproximates $P \in \wp(\mathbb{I}_\Omega)$, that is $P \subseteq \gamma(p)$ then $\alpha(P) \sqsubseteq \alpha(\gamma(p))) \sqsubseteq p$ by (5) and (7) so that $\alpha(P)$ is more precise that $p$ since when considering meanings, $\gamma(\alpha(P)) \subseteq \gamma(q)$. It follows that $\alpha(P)$ is the best overapproximation of $P$ in $L$. The conjunction of properties (4) to (7) is equivalent to

$$\forall P \in \wp(\mathbb{I}_\Omega), p \in L : \alpha(P) \sqsubseteq p \quad\Longleftrightarrow\quad P \subseteq \gamma(p)\ . \tag{8}$$

The above characteristic property (8) of Galois connections is denoted

$$\langle \wp(\mathbb{I}_\Omega), \subseteq\rangle \xleftrightarrows[\alpha]{\gamma} \langle L, \sqsubseteq\rangle\ . \tag{9}$$

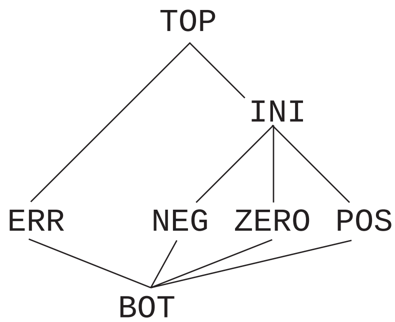



Figure 1: The lattice of initialization and simple signs

Definitions and proofs relative to Galois connections can be found in pages 103–141 of [14] which were distributed to the summer school students as a preliminary introduction to abstract interpretation. Recall that in a Galois connection $\alpha$ preserves existing joins, $\gamma$ preserves existing meets and one adjoint uniquely determine the other. We have

$$\begin{aligned} \alpha(P) &= \sqcap\{p \mid P \subseteq \gamma(p)\}, \\ \gamma(p) &= \cup\{P \mid \alpha(P) \subseteq p\}\ . \end{aligned} \tag{10}$$

It follows that $\alpha(P)$ is the abstract encoding of the concrete property $\gamma(\alpha(P)) = \gamma(\sqcap\{p \mid P \subseteq \gamma(p)\}) = \sqcap\{\gamma(p) \mid P \subseteq \gamma(p)\}$ which is the best overapproximation of the concrete property $P$ by abstract properties $p \in L$ (from above, whence such that $P \subseteq \gamma(p)$).

### 5.2 *Componentwise abstraction of sets of pairs*

The nonrelational/componentwise abstraction of properties of pairs of values (that is sets of pairs) consists in forgetting about the possible relationships between members of these pairs by componentwise application of the Galois connection (9). Formally

$$\alpha^2(P) \triangleq \langle \alpha(\{v_1 \mid \exists v_2 : \langle v_1,\ v_2\rangle \in P\}),\ \alpha(\{v_2 \mid \exists v_1 : \langle v_1,\ v_2\rangle \in P\})\rangle, \tag{11}$$

$$\gamma^2(\langle p_1,\ p_2\rangle) \triangleq \{\langle v_1,\ v_2\rangle \mid v_1 \in \gamma(p_1) \wedge v_2 \in \gamma(p_2)\} \tag{12}$$

so that

$$\langle \wp(\mathbb{I}_\Omega \times \mathbb{I}_\Omega),\ \subseteq\rangle \underset{\alpha^2}{\overset{\gamma^2}{\leftrightarrows}} \langle L \times L,\ \sqsubseteq^2\rangle \tag{13}$$

with the componentwise ordering

$$\langle p_1,\ p_2\rangle \sqsubseteq^2 \langle q_1,\ q_2\rangle \triangleq p_1 \sqsubseteq q_1 \wedge p_2 \sqsubseteq q_2\ .$$

### 5.3 *Initialization and simple sign abstraction*

We now consider an application where abstract properties record initialization and sign only. The lattice $L$ is defined by Hasse diagram of Fig. 1. The meaning of these abstract properties is the following

$$\begin{array}{rcl@{\qquad}rcl} \gamma(\texttt{BOT}) &\triangleq& \{\Omega_{\texttt{a}}\}, & \gamma(\texttt{INI}) &\triangleq& \mathbb{I} \cup \{\Omega_{\texttt{a}}\}, \\ \gamma(\texttt{NEG}) &\triangleq& [\texttt{min\_int}, -1] \cup \{\Omega_{\texttt{a}}\}, & \gamma(\texttt{ERR}) &\triangleq& \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}, \\ \gamma(\texttt{ZERO}) &\triangleq& \{0, \Omega_{\texttt{a}}\}, & \gamma(\texttt{TOP}) &\triangleq& \mathbb{I}_\Omega, \\ \gamma(\texttt{POS}) &\triangleq& [1, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\}\ . & & & \end{array} \tag{14}$$

In order to later illustrate consecutive losses of information, we have chosen not to include the abstract values $\texttt{NEGZ}$, $\texttt{NZERO}$ and $\texttt{POSZ}$ such that $\gamma(\texttt{NEGZ}) \triangleq [\texttt{min\_int}, 0] \cup \{\Omega_{\texttt{a}}\}$, $\gamma(\texttt{NZERO}) \triangleq [\texttt{min\_int}, -1] \cup [1, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\}$ and $\gamma(\texttt{POSZ}) \triangleq [0, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\}$.

Observe that if we had defined $\gamma(\texttt{ERR}) \triangleq \{\Omega_{\texttt{i}}\}$ then $\gamma$ would not be monotone so that (9) would not hold. Another abstract value would be needed to discriminate the initialization and arithmetic errors (see Fig. 3).

Another possible definition of $\gamma$ would have been (14) but with $\gamma(\texttt{BOT}) \triangleq \emptyset$. Then $\gamma$ would not preserve meets (since e.g. $\gamma(\texttt{NEG} \sqcap \texttt{POS}) = \gamma(\texttt{BOT}) = \emptyset \neq \{\Omega_{\texttt{a}}\} = \gamma(\texttt{NEG}) \sqcap \gamma(\texttt{POS})$). It would then follow that $\langle\alpha,\ \gamma\rangle$ is not a Galois connection since best approximations may not exist. For example $\{\Omega_{\texttt{a}}\}$ would be upper approximable by the minimal $\texttt{ERR}$, $\texttt{NEG}$, $\texttt{ZERO}$ or $\texttt{POS}$, none of which being more precise than the others in all contexts.

Another completely different choice of $\gamma$ would be

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
\gamma(\texttt{BOT}) & \triangleq & \emptyset, & \gamma(\texttt{INI}) & \triangleq & \mathbb{I}, \\
\gamma(\texttt{NEG}) & \triangleq & [\texttt{min\_int}, -1], & \gamma(\texttt{ERR}) & \triangleq & \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}, \\
\gamma(\texttt{ZERO}) & \triangleq & \{0\}, & \gamma(\texttt{TOP}) & \triangleq & \mathbb{I}_{\Omega}\ . \\
\gamma(\texttt{POS}) & \triangleq & [1, \texttt{max\_int}], & & &
\end{array}
$$

With such a definition of $\gamma$ for a program analysis taking arithmetic overflows into account, the usual rule of signs $\texttt{POS} + \texttt{POS} = \texttt{POS}$ would not hold since the sums of large positive machine integers may yield an arithmetic error $\Omega_{\texttt{a}}$ such that $\Omega_{\texttt{a}} \not\in \gamma(\texttt{POS})$. The correct version of the rule of sign would be $\texttt{POS} + \texttt{POS} = \texttt{TOP}$, which is too imprecise.

Using (10) and the notation $(c_1\ \textbf{?}\ v_1 \mid c_2\ \textbf{?}\ v_2 \ldots \mid c_n\ \textbf{?}\ v_n\ \textbf{¿}\ v_{n+1})$ to denote $v_1$ when condition $c_1$ holds else $v_2$ when condition $c_2$ holds and so on for $v_n$ or else $v_{n+1}$ when none of the conditions $c_1, \ldots, c_n$ hold, we get the initialization and simple sign abstraction, as follows ($P \in \wp(\mathbb{I}_{\Omega})$)

$$
\begin{array}{rcl}
\alpha(P) & \triangleq & (P \subseteq \{\Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{BOT} \\
& & \mid P \subseteq [\texttt{min\_int}, -1] \cup \{\Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{NEG} \\
& & \mid P \subseteq \{0, \Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{ZERO} \\
& & \mid P \subseteq [1, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{POS} \\
& & \mid P \subseteq \mathbb{I} \cup \{\Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{INI} \\
& & \mid P \subseteq \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}\ \textbf{?}\ \texttt{ERR} \\
& & \textbf{¿}\ \texttt{TOP})\ .
\end{array}
\tag{15}
$$

The adjoined functions $\alpha$ and $\gamma$ satisfy conditions (4) to (7) which are equivalent to the characteristic property (8) of Galois connections (9).

### 5.4 *Initialization and interval abstraction*

The traditional lattice for interval analysis [8, 9] is defined by the Hasse diagram of Fig. 2 (where $-\infty$ and $+\infty$ are either lower and upper bounds of integers or, as considered here, shorthands for $\texttt{max\_int}$ and $\texttt{min\_int}$). The corresponding meaning is

$$
\begin{array}{rcl}
\gamma(\texttt{BOT}) & \triangleq & \emptyset, \\
\gamma([a, b]) & \triangleq & \{x \in \mathbb{I} \mid a \leq x \leq b\}\ .
\end{array}
$$

In order to take initialization and arithmetic errors into account, we can use the lattice with Hasse diagram and concretization function given in Fig. 3. Combining interval and error

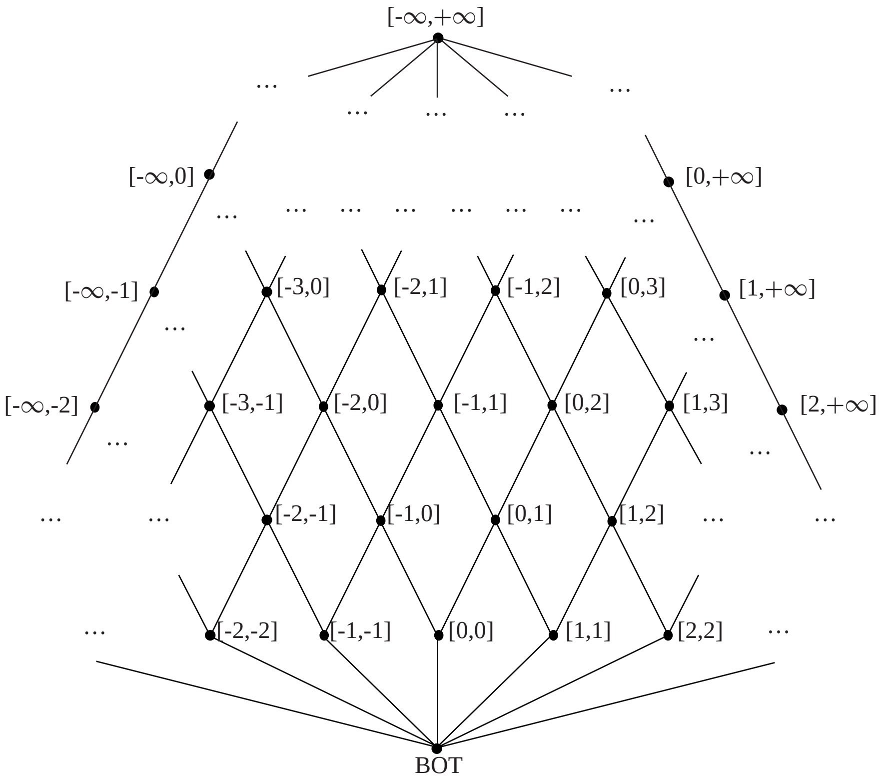
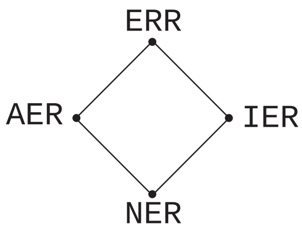



Figure 2: The lattice $I$ of intervals




$$
\begin{aligned}
\gamma(\texttt{NER}) &\triangleq \mathbb{I}\\
\gamma(\texttt{IER}) &\triangleq \mathbb{I} \cup \{\Omega_{\texttt{i}}\}\\
\gamma(\texttt{AER}) &\triangleq \mathbb{I} \cup \{\Omega_{\texttt{a}}\}\\
\gamma(\texttt{ERR}) &\triangleq \mathbb{I} \cup \{\Omega_{\texttt{a}}, \Omega_{\texttt{i}}\}
\end{aligned}
$$

Figure 3: The lattice $E$ of errors

information, we get

$$L \triangleq E \times I$$

with the following meaning

$$\gamma(\langle e,\ i\rangle) \triangleq \gamma(e) \cap \gamma(i)\ .$$

### *5.5 Algebra of abstract properties of values*

The abstract algebra, which consists of abstract values (representing properties of concrete values) and abstract operations (corresponding to abstract property transformers) can be encoded in program modules as follows (the programming language is Objective CAML)

```
module type Abstract_Lattice_Algebra_signature =
  sig
    type lat                              (* abstract properties    *)
    val bot        : unit -> lat          (* infimum                *)
```

```
  val isbotempty : unit -> bool        (* bottom is emptyset?    *)
  val initerr    : unit -> lat         (* uninitialization       *)
  val top        : unit -> lat         (* supremum               *)
  val join       : lat -> lat -> lat   (* least upper bound      *)
  val meet       : lat -> lat -> lat   (* greatest lower bound   *)
  val leq        : lat -> lat -> bool  (* approximation ordering *)
  val eq         : lat -> lat -> bool  (* equality               *)
  val in_errors  : lat -> bool         (* included in errors?    *)
  val print      : lat -> unit
  ...
end;;
```

`(isbotempty ())` is $\gamma(\bot) = \emptyset$ while `(in_errors v)` implies $\gamma(\mathtt{v}) \subseteq \{\Omega_{\mathtt{a}}, \Omega_{\mathtt{i}}\}$.

## 6. Environments

### 6.1 Concrete environments

As usual, we use environments $\rho$ to record the value $\rho(\mathsf{X})$ of program variables $\mathsf{X} \in \mathbb{V}$.

$$\rho \in \mathbb{R} \quad \stackrel{\Delta}{=} \quad \mathbb{V} \mapsto \mathbb{I}_{\Omega}, \qquad \text{environments.}$$

Since environments are functions, we can use the functional assignment/substitution notation defined as ($f \in D \mapsto E$)

$$\begin{aligned} f[d \leftarrow e](x) &\stackrel{\Delta}{=} f(x), \qquad \text{if } x \neq d\ ; \\ f[d \leftarrow e](d) &\stackrel{\Delta}{=} e\ ; \\ f[d_1 \leftarrow e_1; d_2 \leftarrow e_2; \ldots; d_n \leftarrow e_n] &\stackrel{\Delta}{=} (f[d_1 \leftarrow e_1])[d_2 \leftarrow e_2; \ldots; d_n \leftarrow e_n]\ . \end{aligned} \tag{16}$$

### 6.2 Properties of concrete environments

Properties of environments are understood as sets of environments that is elements of $\wp(\mathbb{R})$ where $\subseteq$ is logical implication. Such properties of environments are usually stated using predicates in some prescribed syntactic form. Environment properties are therefore their interpretations. For example the predicate "$\mathsf{X} = \mathsf{Y}$" is interpreted as $\{\rho \in \mathbb{V} \mapsto \mathbb{I}_{\Omega} \mid \rho(\mathsf{X}) = \rho(\mathsf{Y})\}$ and we prefer the second form.

### 6.3 Nonrelational abstraction of environment properties

In order to approximate environment properties, we ignore relationships between the possible values of variables

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_{\Omega}),\ \subseteq \rangle \underset{\alpha_r}{\overset{\gamma_r}{\leftrightarrows}} \langle \mathbb{V} \mapsto \wp(\mathbb{I}_{\Omega}),\ \dot{\subseteq} \rangle$$

by defining

$$\begin{aligned} \alpha_r(R) &= \lambda \mathsf{X} \in \mathbb{V} \cdot \{\rho(\mathsf{X}) \mid \rho \in R\}, \\ \gamma_r(r) &= \{\rho \mid \forall \mathsf{X} \in \mathbb{V} : \rho(\mathsf{X}) \in r(\mathsf{X})\} \end{aligned}$$

and the pointwise ordering which is denoted with the dot notation

$$r \mathrel{\dot{\subseteq}} r' \quad \stackrel{\Delta}{=} \quad \forall \mathsf{X} \in \mathbb{V} : r(\mathsf{X}) \subseteq r'(\mathsf{X})\ .$$

For example if $R = \{[\mathsf{X} \mapsto 1; \mathsf{Y} \mapsto 1], [\mathsf{X} \mapsto 2; \mathsf{Y} \mapsto 2]\}$ then $\alpha_r(R)$ is $[\mathsf{X} \mapsto \{1, 2\}; \mathsf{Y} \mapsto \{1, 2\}]$ so that the equality information ($\mathsf{X} = \mathsf{Y}$) is lost. Since all possible relationships between variables are lost in the nonrelational abstraction, such nonrelational analyzes often lack precision, but are rather efficient.

Now the Galois connection (9)

$$\langle \wp(\mathbb{I}_\Omega),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle$$

can be used to approximate the codomain

$$\langle \mathbb{V} \mapsto \wp(\mathbb{I}_\Omega),\ \dot{\subseteq}\rangle \underset{\alpha_c}{\overset{\gamma_c}{\leftrightarrows}} \langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq}\rangle$$

as follows

$$\begin{aligned}
r \mathrel{\dot{\sqsubseteq}} r' &\stackrel{\Delta}{=} \forall \mathsf{X} \in \mathbb{V} : r(\mathsf{X}) \sqsubseteq r'(\mathsf{X}),\\
\alpha_c(R) &\stackrel{\Delta}{=} \alpha \circ R,\\
\gamma_c(r) &\stackrel{\Delta}{=} \gamma \circ r,
\end{aligned}$$

so that $\langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq},\ \dot{\bot},\ \dot{\top},\ \sqcup,\ \dot{\sqcap}\rangle$ is a complete lattice for the pointwise ordering $\dot{\sqsubseteq}$.

We can now use the fact that the composition of Galois connections

$$\langle L_1,\ \sqsubseteq_1\rangle \underset{\alpha_{21}}{\overset{\gamma_{12}}{\leftrightarrows}} \langle L_2,\ \sqsubseteq_2\rangle \qquad \text{and} \qquad \langle L_2,\ \sqsubseteq_2\rangle \underset{\alpha_{32}}{\overset{\gamma_{23}}{\leftrightarrows}} \langle L_3,\ \sqsubseteq_3\rangle$$

is a Galois connection

$$\langle L_1,\ \sqsubseteq_1\rangle \underset{\alpha_{32}\circ\alpha_{21}}{\overset{\gamma_{12}\circ\gamma_{23}}{\leftrightarrows}} \langle L_3,\ \sqsubseteq_3\rangle\ .$$

The composition of the nonrelational and codomain abstractions is

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega),\ \subseteq\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{V} \mapsto L,\ \dot{\sqsubseteq}\rangle \tag{17}$$

where

$$\begin{aligned}
\dot{\alpha}(R) &\stackrel{\Delta}{=} \alpha_c \circ \alpha_r(R)\\
&= \lambda \mathsf{X} \in \mathbb{V} \centerdot \alpha(\{\rho(\mathsf{X}) \mid \rho \in R\}), &(18)\\
\dot{\gamma}(r) &\stackrel{\Delta}{=} \gamma_r \circ \gamma_c(r)\\
&= \{\rho \mid \forall \mathsf{X} \in \mathbb{V} : \rho(\mathsf{X}) \in \gamma(r(\mathsf{X}))\}\ . &(19)
\end{aligned}$$

If $L$ has an infimum $\bot$ such that $\gamma(\bot) = \emptyset$, we observe that if $r \in \mathbb{V} \mapsto L$ has $\rho(\mathsf{X}) = \bot$ then $\dot{\gamma}(r) = \emptyset$. It follows that the abstract environments with some bottom component all represent the same concrete information ($\emptyset$). The abstract lattice can then be reduced to eliminate equivalent abstract environments (i.e. which have the same meaning) [13, 14]. We have

$$\langle \mathbb{V} \mapsto \mathbb{I}_\Omega,\ \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{V} \overset{\bot}{\longmapsto} L,\ \dot{\sqsubseteq}\rangle \tag{20}$$

where

$$\mathbb{V} \overset{\bot}{\longmapsto} L \ \stackrel{\Delta}{=}\ \{\rho \in \mathbb{V} \mapsto L \mid \forall \mathsf{X} \in \mathbb{V} : \rho(\mathsf{X}) \neq \bot\} \cup \{\lambda \mathsf{X} \in \mathbb{V} \centerdot \bot\}\ .$$

*Numbers*

| | | | |
|---|---|---|---|
| $\mathtt{d} \in \text{Digit}$ | ::= | 0 \| 1 \| ... \| 9 | digits, |
| $\mathtt{n} \in \text{Nat}$ | ::= | Digit \| Nat Digit | numbers in decimal notation. |

*Variables*

| | |
|---|---|
| $\mathtt{X} \in \mathbb{V}$ | variables/identifiers. |

*Arithmetic expressions*

| | | | |
|---|---|---|---|
| $A \in \text{Aexp}$ | ::= | $\mathtt{n}$ | numbers, |
| | \| | $\mathtt{X}$ | variables, |
| | \| | ? | random machine integer, |
| | \| | $+A$ \| $-A$ | unary operators, |
| | \| | $A_1 + A_2$ \| $A_1 - A_2$ | binary operators, |
| | \| | $A_1 * A_2$ \| $A_1 \,/\, A_2$ | |
| | \| | $A_1 \,\mathtt{mod}\, A_2$ . | |

Figure 4: Abstract syntax of arithmetic expressions

### *6.4 Algebra of abstract environments*

In the static analyzer, the complete lattice of environments is encoded by a module parameterized by the module encoding the complete lattice $L$ of abstract properties of values. It is therefore a functor with a formal parameter (along with the expected signature for $L$) which returns the actual structure itself. The static analyzer is *generic* in that by changing the actual parameter one obtains different static analyzers corresponding to different abstractions of properties of values.

```
module type Abstract_Env_Algebra_signature =
  functor (L: Abstract_Lattice_Algebra_signature) ->
  sig
    open Abstract_Syntax
    type env        (* complete lattice of abstract environments   *)
    type element = env
    val bot     : unit -> env            (* infimum                 *)
    val is_bot  : env  -> bool           (* check for infimum       *)
    val initerr : unit -> env            (* uninitialization        *)
    val top     : unit -> env            (* supremum                *)
    val join    : env  -> (env -> env)   (* least upper bound       *)
    val meet    : env  -> (env -> env)   (* greatest lower bound    *)
    val leq     : env  -> (env -> bool)  (* approximation ordering *)
    val eq      : env  -> (env -> bool)  (* equality                *)
    val print   : env  -> unit
    (* substitution *)
    val get     : env  -> variable -> L.lat        (* r(X)          *)
    val set     : env  -> variable -> L.lat -> env (* r[X <- v]     *)
  end;;
```

## 7. Semantics of Arithmetic Expressions

### 7.1 Abstract syntax of arithmetic expressions

The abstract syntax of arithmetic expressions is given in Fig. 4. The random machine integer value ? can be used e.g. to handle inputs of integer variable values.

### 7.2 Machine arithmetics

We respectively write $\underline{\mathtt{n}} \in \mathbb{I}_\Omega$ for the machine natural number and $\mathtt{n} \in \mathbb{N}$ for the mathematical natural number corresponding to the language number $\mathtt{n} \in$ Nat in decimal notation ($\mathtt{d} \in$ Digit, $\mathtt{n} \in$ Nat)

$$\begin{array}{rcll} \underline{\mathtt{d}} & \triangleq & \mathtt{d}; & \\ \underline{\mathtt{nd}} & \triangleq & \Omega_{\mathtt{a}}, & \text{if} \quad 10\,\underline{\mathtt{n}} + \mathtt{d} > \mathtt{max\_int}; \\ \underline{\mathtt{nd}} & \triangleq & 10\,\underline{\mathtt{n}} + \mathtt{d}, & \text{if} \quad 10\,\underline{\mathtt{n}} + \mathtt{d} \leq \mathtt{max\_int}\,. \end{array}$$

We respectively write $\underline{\mathtt{u}} \in \mathbb{I}_\Omega \mapsto \mathbb{I}_\Omega$ for the machine arithmetic operation and $\mathtt{u} \in \mathbb{Z} \mapsto \mathbb{Z}$ for the mathematical arithmetic operation corresponding to the language unary arithmetic operators $\mathtt{u} \in \{+, -\}$. Errors are propagated or raised when the result of the mathematical operation is not machine-representable, so that we have ($e \in \mathbb{E}$, $i \in \mathbb{I}$):

$$\begin{array}{rcll} \underline{\mathtt{u}}\,\Omega_e & \triangleq & \Omega_e; & \\ \underline{\mathtt{u}}\,i & \triangleq & \mathtt{u}\,i, & \text{if} \quad \mathtt{u}\,i \in \mathbb{I}; \\ \underline{\mathtt{u}}\,i & \triangleq & \Omega_{\mathtt{a}}, & \text{if} \quad \mathtt{u}\,i \notin \mathbb{I}\,. \end{array} \tag{21}$$

We respectively write $\underline{\mathtt{b}} \in \mathbb{I}_\Omega \times \mathbb{I}_\Omega \mapsto \mathbb{I}_\Omega$ for the machine arithmetic operation and $\mathtt{b} \in \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{Z}$ for the mathematical arithmetic operation corresponding to the language binary arithmetic operators $\mathtt{b} \in \{+, -, *, /, \mathtt{mod}\}$. Evaluation of operands, whence error propagation is left to right. The division and modulo operations are defined for non-negative first argument and positive second argument. We have ($\mathbb{N}^+$ is the set of positive naturals, $e \in \mathbb{E}$, $v \in \mathbb{I}_\Omega$, $i, i_1, i_2 \in \mathbb{I}$)

$$\begin{array}{rcll} \Omega_e\,\underline{\mathtt{b}}\,v & \triangleq & \Omega_e; & \\ i\,\underline{\mathtt{b}}\,\Omega_e & \triangleq & \Omega_e; & \\ i_1\,\underline{\mathtt{b}}\,i_2 & \triangleq & i_1\,\mathtt{b}\,i_2, & \text{if} \quad \mathtt{b} \in \{+, -, *\} \wedge i_1\,\mathtt{b}\,i_2 \in \mathbb{I}; \\ i_1\,\underline{\mathtt{b}}\,i_2 & \triangleq & i_1\,\mathtt{b}\,i_2, & \text{if} \quad \mathtt{b} \in \{/, \mathtt{mod}\} \wedge i_1 \in \mathbb{I} \cap \mathbb{N} \wedge i_2 \in \mathbb{I} \cap \mathbb{N}^+ \wedge i_1\,\mathtt{b}\,i_2 \in \mathbb{I}; \\ i_1\,\underline{\mathtt{b}}\,i_2 & \triangleq & \Omega_{\mathtt{a}}, & \text{if} \quad i_1\,\mathtt{b}\,i_2 \notin \mathbb{I} \vee (\mathtt{b} \in \{/, \mathtt{mod}\} \wedge (i_1 \notin \mathbb{I} \cap \mathbb{N} \vee i_2 \notin \mathbb{I} \cap \mathbb{N}^+))\,. \end{array} \tag{22}$$

### 7.3 Operational semantics of arithmetic expressions

The big-step operational semantics [31] (renamed natural semantics by [26]) of arithmetic expressions involves judgements $\rho \vdash A \Mapsto v$ meaning that in environment $\rho$, the arithmetic expression $A$ may evaluate to $v \in \mathbb{I}_\Omega$. It is defined in Fig. 5.

[4]Observe that if $\mathtt{m}$ and $\mathtt{M}$ are the strings of digits respectively representing the absolute value of $\mathtt{min\_int}$ and $\mathtt{max\_int}$ then $\underline{\mathtt{m}} > \mathtt{max\_int}$ so that $\rho \vdash \mathtt{m} \Mapsto \Omega_{\mathtt{a}}$ whence $\rho \vdash \mathtt{-\ m} \Mapsto \Omega_{\mathtt{a}}$. However $\rho \vdash \mathtt{(-\ M)\ -\ 1} \Mapsto \mathtt{min\_int}$.

$$\rho \vdash \mathsf{n} \Rightarrow \underline{\mathsf{n}}, \qquad \text{decimal numbers;} \tag{23}$$

$$\rho \vdash \mathsf{X} \Rightarrow \rho(\underline{\mathsf{X}}), \qquad \text{variables;} \tag{24}$$

$$\frac{i \in \mathbb{I}}{\rho \vdash \mathsf{?} \Rightarrow i}, \qquad \text{random;} \tag{25}$$

$$\frac{\rho \vdash A \Rightarrow v}{\rho \vdash \mathsf{u}\, A \Rightarrow \underline{\mathsf{u}}\, v}, \qquad \text{unary arithmetic operations;}^{4} \tag{26}$$

$$\frac{\rho \vdash A_1 \Rightarrow v_1,\ \rho \vdash A_2 \Rightarrow v_2}{\rho \vdash A_1\, \mathsf{b}\, A_2 \Rightarrow v_1\, \underline{\mathsf{b}}\, v_2}, \qquad \text{binary arithmetic operations .} \tag{27}$$

Figure 5: Operational semantics of arithmetic expressions

### 7.4 *Forward collecting semantics of arithmetic expressions*

The *forward/bottom-up collecting semantics* of an arithmetic expression defines the possible values that the arithmetic expression can evaluate to in a given set of environments

$$\begin{aligned} \mathrm{Faexp} &\in \mathrm{Aexp} \mapsto \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{I}_\Omega), \\ \mathrm{Faexp}[\![A]\!]R &\triangleq \{v \mid \exists \rho \in R : \rho \vdash A \Rightarrow v\} . \end{aligned} \tag{28}$$

The forward collecting semantics $\mathrm{Faexp}[\![A]\!]R$ specifies the strongest postcondition that values of the arithmetic expression $A$ do satisfy when this expression is evaluated in an environment satisfying the precondition $R$. The forward collecting semantics can therefore be understood as a predicate transformer [22]. In particular it is a complete join morphism (denoted with $\xrightarrow{\text{cjm}}$), that is ($\mathcal{S}$ is an arbitrary set)

$$\mathrm{Faexp}[\![A]\!]\Big(\bigcup_{k\in\mathcal{S}} R_k\Big) = \bigcup_{k\in\mathcal{S}} \big(\mathrm{Faexp}[\![A]\!]R_k\big) ,$$

which implies monotony (when $\mathcal{S} = \{1, 2\}$ and $R_1 \subseteq R_2$) and $\emptyset$-strictness (when $\mathcal{S} = \emptyset$)

$$\mathrm{Faexp}[\![A]\!]\emptyset = \emptyset .$$

### 7.5 *Backward collecting semantics of arithmetic expressions*

The *backward/top-down collecting semantics* $\mathrm{Baexp}[\![A]\!](R)P$ of an arithmetic expression $A$ defines the subset of possible environments $R$ such that the arithmetic expression may evaluate, without producing a runtime error, to a value belonging to given set $P$

$$\begin{aligned} \mathrm{Baexp} &\in \mathrm{Aexp} \mapsto \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{I}_\Omega) \xrightarrow{\text{cjm}} \wp(\mathbb{R}), \\ \mathrm{Baexp}[\![A]\!](R)P &\triangleq \{\rho \in R \mid \exists i \in P \cap \mathbb{I} : \rho \vdash A \Rightarrow i\} . \end{aligned} \tag{29}$$

## 8. Abstract Interpretation of Arithmetic Expressions

### 8.1 *Lifting Galois connections at higher-order*

In order to approximate monotonic predicate transformers knowing an abstraction (9) of value properties and (20) of environment properties, we use the following functional abstraction [13]

$$\begin{aligned} \alpha^{\triangleright}(\Phi) &\stackrel{\Delta}{=} \alpha \circ \Phi \circ \dot{\gamma}, \\ \gamma^{\triangleright}(\varphi) &\stackrel{\Delta}{=} \gamma \circ \varphi \circ \dot{\alpha} \end{aligned} \tag{30}$$

so that

$$\langle \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega) \xrightarrow{\text{mon}} \wp(\mathbb{I}_\Omega),\ \dot{\subseteq}\rangle \underset{\alpha^{\triangleright}}{\overset{\gamma^{\triangleright}}{\leftrightarrows}} \langle (\mathbb{V} \mapsto L) \xrightarrow{\text{mon}} L,\ \dot{\sqsubseteq}\rangle\ . \tag{31}$$

The intuition is that for any abstract precondition $p \in L$, or its concrete equivalent $\dot{\gamma}(p) \in \wp(\mathbb{V} \mapsto \mathbb{I}_\Omega)$, the abstract predicate transformer $\varphi$ should provide an overestimate $\varphi(p)$ of the postcondition $\Phi(\gamma(p))$ defined by the concrete predicate transformer $\Phi$. This soundness requirement can be formalized as follows:

$$\begin{array}{lll} & \forall p \in L : \gamma(\varphi(p)) \supseteq \Phi(\dot{\gamma}(p)) & \wr\text{soundness requirement}\wr \\ \Longleftrightarrow & \forall p \in L : \Phi(\dot{\gamma}(p)) \subseteq \gamma(\varphi(p)) & \wr\text{def. inverse } \supseteq \text{ of } \subseteq\wr \\ \Longleftrightarrow & \forall p \in L : \alpha(\Phi(\dot{\gamma}(p))) \sqsubseteq \varphi(p) & \wr\text{def. Galois connection}\wr \\ \Longleftrightarrow & \alpha \circ \Phi \circ \dot{\gamma} \mathrel{\dot{\sqsubseteq}} \varphi & \wr\text{def. } \dot{\sqsubseteq}\wr \\ \Longleftrightarrow & \alpha^{\triangleright}(\Phi) \mathrel{\dot{\sqsubseteq}} \varphi & \wr\text{def. } \alpha^{\triangleright}\wr. \end{array} \tag{32}$$

Choosing $\varphi \stackrel{\Delta}{=} \alpha^{\triangleright}(\Phi)$ is therefore the best of the possible sound choices since it always provides the strongest abstract postcondition, whence, by monotony, the strongest concrete one.

Observe that $\Phi$ (as defined by the collecting semantics) and $\alpha$ are (in general) not computable so that $\alpha^{\triangleright}(\Phi)$ was not proposed by [13] as an implementation of the abstract predicate transformer but instead as a formal specification. In practice, this specification must be refined into an algorithm effectively computing the abstract predicate transformer $\varphi$. This point is sometimes misunderstood [28].

Moreover [13] does not require the abstract predicate transformer $\varphi$ to be chosen as the best possible choice $\alpha^{\triangleright}(\Phi)$. Clearly (32) shows that any overestimate is sound (although less precise but hopefully more efficiently computable). This is also sometimes misunderstood [28].

### 8.2 *Generic forward/top-down abstract interpretation of arithmetic expressions*

We now design the generic forward/top-down nonrelational abstract semantics of arithmetic expressions

$$\begin{array}{lll} \text{Faexp}^{\triangleright} \in \text{Aexp} \mapsto (\mathbb{V} \longmapsto L) \xrightarrow{\text{mon}} L, & \text{when} & \gamma(\bot) \neq \emptyset; \\ \text{Faexp}^{\triangleright} \in \text{Aexp} \mapsto (\mathbb{V} \overset{\bot}{\longmapsto} L) \xrightarrow{\text{mon}} L, & \text{when} & \gamma(\bot) = \emptyset \end{array}$$

by calculus. This consists consists, for any possible approximation (9) of value properties, in approximating environment properties by the nonrelational abstraction (20) and in applying the functional abstraction (31) to the forward collecting semantics (28). We get an overapproximation such that

$$\mathrm{Faexp}^{\triangleright}[\![A]\!] \;\;\dot{\sqsupseteq}\;\; \alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])\ . \tag{33}$$

Starting from the formal specification $\alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])$, we derive an algorithm $\mathrm{Faexp}^{\triangleright}[\![A]\!]$ satisfying (33) by calculus

$$\begin{array}{ll}
 & \alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!]) \\
= & \quad \wr\text{def. (30) of } \alpha^{\triangleright}\wr \\
 & \alpha \circ \mathrm{Faexp}[\![A]\!] \circ \dot{\gamma} \\
= & \quad \wr\text{def. of composition } \circ\wr \\
 & \lambda r \bullet \alpha(\mathrm{Faexp}[\![A]\!](\dot{\gamma}(r))) \\
= & \quad \wr\text{def. (28) of } \mathrm{Faexp}[\![A]\!]\wr \\
 & \lambda r \bullet \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Mapsto v\})\ .
\end{array}$$

If $r$ is the infimum $\lambda \mathsf{Y} \bullet \bot$ where the infimum $\bot$ of $L$ is such that $\gamma(\bot) = \emptyset$, then $\dot{\gamma}(r) = \emptyset$ whence:

$$\begin{array}{ll}
 & \alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])(\lambda \mathsf{Y} \bullet \bot) \\
= & \quad \wr\text{def. (19) of } \dot{\gamma}\wr \\
 & \alpha(\emptyset) \\
= & \quad \wr\text{Galois connection (9) so that } \alpha(\emptyset) = \bot\wr \\
 & \bot\ .
\end{array}$$

When $r \neq \lambda \mathsf{Y} \bullet \bot$ or $\gamma(\bot) \neq \emptyset$, we have

$$\begin{array}{ll}
 & \alpha^{\triangleright}(\mathrm{Faexp}[\![A]\!])r \\
= & (\lambda r \bullet \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Mapsto v\}))r \\
= & \quad \wr\text{def. lambda expression}\wr \\
 & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A \Mapsto v\})
\end{array}$$

and we proceed by induction on the arithmetic expression $A$.

1 — When $A = \mathsf{n} \in \mathrm{Nat}$ is a number, we have

$$\begin{array}{ll}
 & \alpha^{\triangleright}(\mathrm{Faexp}[\![\mathsf{n}]\!])r \\
= & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash \mathsf{n} \Mapsto v\}) \\
= & \quad \wr\text{def. (23) of } \rho \vdash \mathsf{n} \Mapsto v\wr \\
 & \alpha(\{\underline{\mathsf{n}}\}) \\
= & \quad \wr\text{by defining } \mathsf{n} = \alpha(\{\underline{\mathsf{n}}\})\wr \\
 & \mathsf{n} \\
= & \quad \wr\text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathsf{n}]\!]r \triangleq \mathsf{n}\wr \\
 & \mathrm{Faexp}^{\triangleright}[\![\mathsf{n}]\!]r\ .
\end{array}$$

2 — When $A = \mathsf{X} \in \mathbb{V}$ is a variable, we have

$$\begin{array}{ll}
 & \alpha^{\triangleright}(\mathrm{Faexp}[\![\mathsf{X}]\!])r \\
= & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash \mathsf{X} \Mapsto v\}) \\
= & \quad \wr\text{def. (24) of } \rho \vdash \mathsf{X} \Mapsto v\wr \\
 & \alpha(\{\rho(\underline{\mathsf{X}}) \mid \rho \in \dot{\gamma}(r)\}) \\
= & \quad \wr\text{def. (19) of } \dot{\gamma}\wr
\end{array}$$

$$
\begin{array}{cl}
& \alpha(\gamma(r(\mathsf{X}))) \\
\sqsubseteq & \quad \wr \text{Galois connection (9) so that } \alpha \circ \gamma \text{ is reductive} \wr \\
& r(\mathsf{X}) \\
= & \quad \wr \text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathsf{X}]\!]r \triangleq r(\mathsf{X}) \wr \\
& \mathrm{Faexp}^{\triangleright}[\![\mathsf{X}]\!]r \ .
\end{array}
$$

3 — When $A = ?$ is random, we have

$$
\begin{array}{cl}
& \alpha^{\triangleright}(\mathrm{Faexp}[\![?]\!])r \\
= & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash ? \Rightarrow v\}) \\
= & \quad \wr \text{def. (25) of } \rho \vdash ? \Rightarrow v \wr \\
& \alpha(\mathbb{I}) \\
\sqsubseteq & \quad \wr \text{by defining } ?^{\triangleright} \sqsupseteq \alpha(\mathbb{I}) \wr \\
& ?^{\triangleright} \\
= & \quad \wr \text{by defining } \mathrm{Faexp}^{\triangleright}[\![?]\!]r \triangleq ?^{\triangleright} \wr \\
& \mathrm{Faexp}^{\triangleright}[\![?]\!]r \ .
\end{array}
$$

4 — When $A = \mathsf{u}\ A'$ is a unary operation, we have

$$
\begin{array}{cl}
& \alpha^{\triangleright}(\mathrm{Faexp}[\![\mathsf{u}\ A']\!])r \\
= & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash \mathsf{u}\ A' \Rightarrow v\}) \\
= & \quad \wr \text{def. (4) of } \rho \vdash \mathsf{u}\ A' \Rightarrow v \wr \\
& \alpha(\{\underline{\mathsf{u}}\, v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A' \Rightarrow v\}) \\
\sqsubseteq & \quad \wr \gamma \circ \alpha \text{ is extensive (6), } \alpha \text{ is monotone (5)} \wr \\
& \alpha(\{\underline{\mathsf{u}}\, v \mid v \in \gamma \circ \alpha(\{v' \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A' \Rightarrow v'\})\}) \\
\sqsubseteq & \quad \wr \text{induction hypothesis (33), } \gamma \text{ (4) and } \alpha \text{ (5) are monotone} \wr \\
& \alpha(\{\underline{\mathsf{u}}\, v \mid v \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A']\!]r)\}) \\
\sqsubseteq & \quad \wr \text{by defining } \mathsf{u}^{\triangleright} \text{ such that } \mathsf{u}^{\triangleright}(p) \sqsupseteq \alpha(\{\underline{\mathsf{u}}\, v \mid v \in \gamma(p)\}) \wr \\
& \mathsf{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r) \\
= & \quad \wr \text{by defining } \mathrm{Faexp}^{\triangleright}[\![\mathsf{u}\ A']\!]r \triangleq \mathsf{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r) \wr \\
& \mathrm{Faexp}^{\triangleright}[\![\mathsf{u}\ A']\!]r \ .
\end{array}
$$

5 — When $A = A_1\ \mathsf{b}\ A_2$ is a binary operation, we have

$$
\begin{array}{cl}
& \alpha^{\triangleright}(\mathrm{Faexp}[\![A_1\ \mathsf{b}\ A_2]\!])r \\
= & \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_1\ \mathsf{b}\ A_2 \Rightarrow v\}) \\
= & \quad \wr \text{def. (27) of } \rho \vdash A_1\ \mathsf{b}\ A_2 \Rightarrow v \wr \\
& \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_1 \Rightarrow v_1 \wedge \rho \vdash A_2 \Rightarrow v_2\}) \\
\sqsubseteq & \quad \wr \alpha \text{ monotone (5)} \wr \\
& \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid \exists \rho_1 \in \dot{\gamma}(r) : \rho_1 \vdash A_1 \Rightarrow v_1 \wedge \exists \rho_2 \in \dot{\gamma}(r) : \rho_2 \vdash A_2 \Rightarrow v_2\}) \\
\sqsubseteq & \quad \wr \gamma \circ \alpha \text{ is extensive (6), } \alpha \text{ is monotone (5)} \wr \\
& \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid v_1 \in \gamma \circ \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_1 \Rightarrow v\}) \wedge \\
& \qquad\qquad v_2 \in \gamma \circ \alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_2 \Rightarrow v\})\}) \\
\sqsubseteq & \quad \wr \text{induction hypothesis (33), } \gamma \text{ (4) and } \alpha \text{ (5) are monotone} \wr \\
& \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid v_1 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r) \wedge v_2 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_2]\!]r)\}) \\
\sqsubseteq & \quad \wr \text{by defining } \mathsf{b}^{\triangleright} \text{ such that } \mathsf{b}^{\triangleright}(p_1, p_2) \sqsupseteq \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid v_1 \in \gamma(p_1) \wedge v_2 \in \gamma(p_2)\}) \wr \\
& \mathsf{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \\
= & \quad \wr \text{by defining } \mathrm{Faexp}^{\triangleright}[\![A_1\ \mathsf{b}\ A_2]\!]r \triangleq \mathsf{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \wr \\
& \mathrm{Faexp}^{\triangleright}[\![A_1\ \mathsf{b}\ A_2]\!]r \ .
\end{array}
$$

$$
\begin{array}{rcll}
\mathrm{Faexp}^{\triangleright}[\![A]\!](\lambda \mathsf{Y}\bullet\bot) & \stackrel{\Delta}{=} & \bot & \text{if}\quad \gamma(\bot)=\emptyset \qquad (34)\\
\mathrm{Faexp}^{\triangleright}[\![\mathsf{n}]\!]r & \stackrel{\Delta}{=} & \mathsf{n}^{\triangleright} & \\
\mathrm{Faexp}^{\triangleright}[\![\mathsf{X}]\!]r & \stackrel{\Delta}{=} & r(\mathsf{X}) & \\
\mathrm{Faexp}^{\triangleright}[\![?]\!]r & \stackrel{\Delta}{=} & ?^{\triangleright} & \\
\mathrm{Faexp}^{\triangleright}[\![\mathsf{u}\ A']\!]r & \stackrel{\Delta}{=} & \mathsf{u}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A']\!]r) & \\
\mathrm{Faexp}^{\triangleright}[\![A_1\ \mathsf{b}\ A_2]\!]r & \stackrel{\Delta}{=} & \mathsf{b}^{\triangleright}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) &
\end{array}
$$

parameterized by the following forward abstract operations

$$
\begin{array}{rclrcll}
\mathsf{n}^{\triangleright} & = & \alpha(\{\underline{\mathsf{n}}\}) & \mathsf{u}^{\triangleright}(p) & \sqsupseteq & \alpha(\{\underline{\mathsf{u}}\, v \mid v\in\gamma(p)\}) & (35)\\
?^{\triangleright} & \sqsupseteq & \alpha(\mathbb{I}) & \mathsf{b}^{\triangleright}(p_1,p_2) & \sqsupseteq & \alpha(\{v_1\ \underline{\mathsf{b}}\ v_2 \mid v_1\in\gamma(p_1)\wedge v_2\in\gamma(p_2)\}) & (36)
\end{array}
$$

Figure 6: Forward abstract interpretation of arithmetic expressions

In conclusion, we have designed the forward abstract interpretation $\mathrm{Faexp}^{\triangleright}$ of arithmetic expressions in such a way that satisfies the soundness requirement (33) as summarized in Fig. 6.

By structural induction on the arithmetic expression $A$, the abstract semantics $\mathrm{Faexp}^{\triangleright}[\![A]\!]$ of $A$ is monotonic (respectively continuous) if the abstract operations $\mathsf{u}^{\triangleright}$ and $\mathsf{b}^{\triangleright}$ are monotonic (resp. continuous), since the composition of monotonic (resp. continuous) functions is monotonic (resp. continuous).

### *8.3 Generic forward/top-down static analyzer of arithmetic expressions*

The operations on abstract value properties which are used for the forward abstract interpretation of arithmetic expressions of Fig. 6 must be provided with the module implementing each particular algebra of abstract properties.

```
module type Abstract_Lattice_Algebra_signature =
  sig
    (* complete lattice of abstract properties of values         *)
    type lat                                 (* abstract properties    *)
    ...
    (* forward abstract interpretation of arithmetic expressions *)
    val f_INT    : string -> lat
    val f_RANDOM : unit -> lat
    val f_UMINUS : lat -> lat
    val f_UPLUS  : lat -> lat
    val f_PLUS   : lat -> lat -> lat
    val f_MINUS  : lat -> lat -> lat
    val f_TIMES  : lat -> lat -> lat
    val f_DIV    : lat -> lat -> lat
    val f_MOD    : lat -> lat -> lat
    ...
  end;;
```

In functional programming, the translation from Fig. 6 to a program is immediate as follows

```
module type Faexp_signature =
```

```
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  sig
    open Abstract_Syntax
    (* generic forward abstract interpretation of arithmetic operations *)
    val faexp : aexp ->  E(L).env -> L.lat
  end;;

module Faexp_implementation =
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  struct
    open Abstract_Syntax
    (* generic abstract environments *)
    module E'=E(L)
    (* generic forward abstract interpretation of arithmetic operations *)
    let rec faexp' a r =
      match a with
      | (INT i)          -> (L.f_INT i)
      | (VAR v)          -> (E'.get r v)
      | RANDOM           -> (L.f_RANDOM ())
      | (UMINUS a1)      -> (L.f_UMINUS (faexp' a1 r))
      | (UPLUS a1)       -> (L.f_UPLUS (faexp' a1 r))
      | (PLUS (a1, a2))  -> (L.f_PLUS (faexp' a1 r) (faexp' a2 r))
      | (MINUS (a1, a2)) -> (L.f_MINUS (faexp' a1 r) (faexp' a2 r))
      | (TIMES (a1, a2)) -> (L.f_TIMES (faexp' a1 r) (faexp' a2 r))
      | (DIV (a1, a2))   -> (L.f_DIV (faexp' a1 r) (faexp' a2 r))
      | (MOD (a1, a2))   -> (L.f_MOD (faexp' a1 r) (faexp' a2 r))
    let faexp a r =
      if (E'.is_bot r) & (L.isbotempty ()) then (L.bot ()) else faexp' a r
  end;;

module Faexp = (Faexp_implementation:Faexp_signature);;
```

Speed and low memory consumption are definitely required for analyzing very large programs. This may require a much more efficient implementation where the abstract interpreter [7] is replaced by an abstract compiler producing code for each arithmetic expression to be analyzed using may be register allocation algorithms and why not common subexpressions elimination (see e.g. Ch. 9.10 of [1]) to minimize the number of intermediate abstract environments to be allocated and deallocated.

### *8.4 Initialization and simple sign abstract forward arithmetic operations*

Considering the initialization and simple sign abstraction of Sec. 5.3, the calculational design of the forward abstract operations proceeds as follows

$$
\begin{aligned}
& \alpha(\{\underline{\mathtt{n}}\}) \\
= \quad & \wr(15) \text{ and case analysis} \wr \\
& \begin{array}{ll}
\mathtt{NEG} & \text{if } \ \underline{\mathtt{n}} \in [\mathtt{min\_int}, -1] \\
\mathtt{ZERO} & \text{if } \ \underline{\mathtt{n}} = 0 \\
\mathtt{POS} & \text{if } \ \underline{\mathtt{n}} \in [1, \mathtt{max\_int}] \\
\mathtt{BOT} & \text{if } \ \underline{\mathtt{n}} < \mathtt{min\_int} \ \text{ or } \ \underline{\mathtt{n}} > \mathtt{max\_int}
\end{array} \\
\stackrel{\Delta}{=} \ & \mathtt{n}^{\triangleright} \ .
\end{aligned}
$$

$$\begin{aligned}
& \alpha(\mathbb{I}) \\
= & \quad \wr(15)\wr \\
& \texttt{INI} \\
\triangleq & \; ?^{\triangleright} \; .
\end{aligned}$$

We design $-^{\triangleright}(p) \triangleq \alpha(\{- v \mid v \in \gamma(p)\})$ by case analysis

$$\begin{array}{lcll}
-^{\triangleright}(\texttt{BOT}) & = & \alpha(\{- v \mid v \in \gamma(\texttt{BOT})\}) & \wr\text{def. (35) of } -^{\triangleright}\wr \\
& = & \alpha(\{\underline{-}\, v \mid v \in \{\Omega_{\texttt{a}}\}\}) & \wr\text{def. (14) of } \gamma\wr \\
& = & \alpha(\{\Omega_{\texttt{a}}\}) & \wr\text{def. (21) of } \underline{-}\wr \\
& = & \texttt{BOT} & \wr\text{def. (15) of } \alpha\wr \\
-^{\triangleright}(\texttt{POS}) & = & \alpha(\{- v \mid v \in \gamma(\texttt{POS})\}) & \wr\text{def. (35) of } -^{\triangleright}\wr \\
& = & \alpha(\{\underline{-}\, v \mid v \in [1, \texttt{max_int}] \cup \{\Omega_{\texttt{a}}\}\}) & \wr\text{def. (14) of } \gamma\wr \\
& = & \alpha([-\texttt{max_int}, -1] \cup \{\Omega_{\texttt{a}}\}) & \wr\text{def. (21) of } \underline{-} \text{ and (2)}\wr \\
& = & \texttt{NEG} & \wr\text{def. (15) of } \alpha\wr \\
-^{\triangleright}(\texttt{ERR}) & = & \alpha(\{- v \mid v \in \gamma(\texttt{ERR})\}) & \wr\text{def. (35) of } -^{\triangleright}\wr \\
& = & \alpha(\{\underline{-}\, v \mid v \in \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}\}) & \wr\text{def. (14) of } \gamma\wr \\
& = & \alpha(\{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}) & \wr\text{def. (21) of } \underline{-}\wr \\
& = & \texttt{ERR} & \wr\text{def. (15) of } \alpha\wr
\end{array}$$

The calculational design for the other cases of $-^{\triangleright}$ and that of $+^{\triangleright}$ is similar and we get

| $p$ | BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|
| $+^{\triangleright}(p)$ | BOT | NEG | ZERO | POS | INI | ERR | TOP |
| $-^{\triangleright}(p)$ | BOT | POS | ZERO | NEG | INI | ERR | TOP |

The calculational design of the abstract binary operators is also similar and will not be fully detailed. For division, we get

| | $/^{\triangleright}(p, q)$ | $q$: BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|---|
| $p$ | BOT | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | NEG | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | ZERO | BOT | BOT | BOT | ZERO | POS | ERR | TOP |
| | POS | BOT | BOT | BOT | INI | INI | ERR | TOP |
| | INI | BOT | BOT | BOT | INI | INI | ERR | TOP |
| | ERR | ERR | ERR | ERR | ERR | ERR | ERR | ERR |
| | TOP | ERR | ERR | ERR | TOP | TOP | ERR | TOP |

Let us consider a few typical cases. First division by a negative number always leads to an arithmetic error

$$\begin{array}{lcll}
/^{\triangleright}(\texttt{POS}, \texttt{NEG}) & = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \gamma(\texttt{POS}) \wedge v_2 \in \gamma(\texttt{NEG})\}) & \wr\text{def. (36) of } /^{\triangleright}\wr \\
& = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in [1, \texttt{max_int}] \cup \{\Omega_{\texttt{a}}\} \wedge & \wr\text{def. (14) of } \gamma\wr \\
& & \qquad v_2 \in [\texttt{min_int}, -1] \cup \{\Omega_{\texttt{a}}\}\}) & \\
& = & \alpha(\{\Omega_{\texttt{a}}\}) & \wr\text{def. (22) of } \underline{/}\wr \\
& = & \texttt{BOT} & \wr\text{def. (15) of } \alpha\wr
\end{array}$$

No abstract property exactly represents non-negative numbers which yields imprecise results

$$
\begin{array}{rcll}
/^{\triangleright}(\mathtt{POS}, \mathtt{POS}) & = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{POS}) \wedge v_2 \in \gamma(\mathtt{POS})\}) & \wr \text{def. (36) of } /^{\triangleright} \wr \\
& = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathtt{a}}\} \wedge & \wr \text{def. (14) of } \gamma \wr \\
& & \qquad v_2 \in [1, \mathtt{max\_int}] \cup \{\Omega_{\mathtt{a}}\})\}) & \\
& = & \alpha([0, \mathtt{max\_int}] \cup \{\Omega_{\mathtt{a}}\}) & \wr \text{def. (22) of } \underline{/} \wr \\
& = & \mathtt{INI} & \wr \text{def. (15) of } \alpha \wr
\end{array}
$$

Because of left to right evaluation, left errors are propagated first

$$
\begin{array}{rcll}
/^{\triangleright}(\mathtt{BOT}, \mathtt{ERR}) & = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{BOT}) \wedge v_2 \in \gamma(\mathtt{ERR})\}) & \wr \text{def. (36) of } /^{\triangleright} \wr \\
& = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \{\Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \wr \text{def. (14) of } \gamma \wr \\
& = & \alpha(\{\Omega_{\mathtt{a}}\}) & \wr \text{def. (22) of } \underline{/} \wr \\
& = & \mathtt{BOT} & \wr \text{def. (15) of } \alpha \wr \\
/^{\triangleright}(\mathtt{ERR}, \mathtt{BOT}) & = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{ERR}) \wedge v_2 \in \gamma(\mathtt{BOT})\}) & \wr \text{def. (36) of } /^{\triangleright} \wr \\
& = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{a}}\}) & \wr \text{def. (14) of } \gamma \wr \\
& = & \alpha(\{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \wr \text{def. (22) of } \underline{/} \wr \\
& = & \mathtt{ERR} & \wr \text{def. (15) of } \alpha \wr \\
/^{\triangleright}(\mathtt{TOP}, \mathtt{BOT}) & = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in \gamma(\mathtt{TOP}) \wedge v_2 \in \gamma(\mathtt{BOT})\}) & \wr \text{def. (36) of } /^{\triangleright} \wr \\
& = & \alpha(\{v_1 \,\underline{/}\, v_2 \mid v_1 \in [\mathtt{min\_int}, \mathtt{max\_int}] \cup & \wr \text{def. (14) of } \gamma \wr \\
& & \qquad \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\} \wedge v_2 \in \{\Omega_{\mathtt{a}}\}) & \\
& = & \alpha(\{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}) & \wr \text{def. (22) of } \underline{/} \wr \\
& = & \mathtt{ERR} & \wr \text{def. (15) of } \alpha \wr
\end{array}
$$

The other forward abstract binary arithmetic operators for initialization and simple sign analysis are as follows

| | $+^{\triangleright}(p, q)$ | $q$: BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|---|
| $p$ | BOT | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | NEG | BOT | NEG | NEG | INI | INI | ERR | TOP |
| | ZERO | BOT | NEG | ZERO | POS | INI | ERR | TOP |
| | POS | BOT | INI | POS | POS | INI | ERR | TOP |
| | INI | BOT | INI | INI | INI | INI | ERR | TOP |
| | ERR | ERR | ERR | ERR | ERR | ERR | ERR | ERR |
| | TOP | ERR | TOP | TOP | TOP | TOP | ERR | TOP |

| | $-^{\triangleright}(p, q)$ | $q$: BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|---|
| $p$ | BOT | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | NEG | BOT | INI | NEG | NEG | INI | ERR | TOP |
| | ZERO | BOT | POS | ZERO | NEG | INI | ERR | TOP |
| | POS | BOT | POS | POS | INI | INI | ERR | TOP |
| | INI | BOT | INI | INI | INI | INI | ERR | TOP |
| | ERR | ERR | ERR | ERR | ERR | ERR | ERR | ERR |
| | TOP | ERR | TOP | TOP | TOP | TOP | ERR | TOP |

| | $*^{\triangleright}(p, q)$ | $q$: BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|---|
| $p$ | BOT | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | NEG | BOT | POS | ZERO | NEG | INI | ERR | TOP |
| | ZERO | BOT | ZERO | ZERO | ZERO | ZERO | ERR | TOP |
| | POS | BOT | NEG | ZERO | POS | INI | ERR | TOP |
| | INI | BOT | INI | ZERO | INI | INI | ERR | TOP |
| | ERR | ERR | ERR | ERR | ERR | ERR | ERR | ERR |
| | TOP | ERR | TOP | TOP | TOP | TOP | ERR | TOP |

| | $\mathtt{mod}^{\triangleright}(p, q)$ | $q$: BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|---|
| $p$ | BOT | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | NEG | BOT | BOT | BOT | BOT | BOT | BOT | BOT |
| | ZERO | BOT | BOT | BOT | ZERO | ZERO | ERR | TOP |
| | POS | BOT | BOT | BOT | INI | INI | ERR | TOP |
| | INI | BOT | BOT | BOT | INI | INI | ERR | TOP |
| | ERR | ERR | ERR | ERR | ERR | ERR | ERR | ERR |
| | TOP | ERR | ERR | ERR | TOP | TOP | ERR | TOP |

### 8.5 *Generic backward/bottom-up abstract interpretation of arithmetic expressions*

We now design the backward/bottom-up abstract semantics of arithmetic expressions

$$
\mathrm{Baexp}^{\triangleleft} \;\in\; \mathrm{Aexp} \mapsto (\mathbb{V} \mapsto L) \stackrel{\text{mon}}{\longmapsto} L \stackrel{\text{mon}}{\longmapsto} (\mathbb{V} \mapsto L)\ .
$$

For any possible approximation (9) of value properties, we approximate environment properties by the nonrelational abstraction (20) and apply the following functional abstraction

$$
\langle \wp(\mathbb{R}) \stackrel{\text{mon}}{\longmapsto} \wp(\mathbb{I}_{\Omega}) \stackrel{\text{mon}}{\longmapsto} \wp(\mathbb{R}),\ \ddot{\subseteq} \rangle \xleftarrow[\alpha^{\triangleleft}]{\gamma^{\triangleleft}} \langle (\mathbb{V} \mapsto L) \stackrel{\text{mon}}{\longmapsto} L \stackrel{\text{mon}}{\longmapsto} (\mathbb{V} \mapsto L),\ \ddot{\sqsubseteq} \rangle
$$

where

$$
\begin{array}{rcl}
\Phi \mathrel{\ddot{\subseteq}} \Psi & \triangleq & \forall R \in \wp(\mathbb{R}) : \forall P \in \wp(\mathbb{I}_\Omega) : \Phi(R)P \subseteq \Psi(R)P, \\
\varphi \mathrel{\ddot{\sqsubseteq}} \psi & \triangleq & \forall r \in \mathbb{V} \mapsto L : \forall p \in L : \varphi(r)p \mathrel{\dot{\sqsubseteq}} \psi(r)p, \\
\alpha^{\triangleleft}(\Phi) & \triangleq & \lambda r \in \mathbb{V} \mapsto L \cdot \lambda p \in L \cdot \dot{\alpha}(\Phi(\dot{\gamma}(r))\gamma(p)), \\
\gamma^{\triangleleft}(\varphi) & \triangleq & \lambda R \in \wp(\mathbb{R}) \cdot \lambda P \in \wp(\mathbb{I}_\Omega) \cdot \dot{\gamma}(\varphi(\dot{\alpha}(R))\alpha(P)) \ .
\end{array}
\tag{37}
$$

The objective is to get an overapproximation of the backward collecting semantics (29) such that

$$
\mathrm{Baexp}^{\triangleleft}[\![A]\!] \mathrel{\ddot{\sqsupseteq}} \alpha^{\triangleleft}(\mathrm{Baexp}[\![A]\!]) \ . \tag{38}
$$

We derive $\mathrm{Baexp}^{\triangleleft}[\![A]\!]$ by calculus, as follows

$$
\begin{array}{ll}
 & \alpha^{\triangleleft}(\mathrm{Baexp}[\![A]\!]) \\
= & \quad \wr \text{def. (37) of } \alpha^{\triangleleft} \int \\
 & \lambda r \in \mathbb{V} \mapsto L \cdot \lambda p \in L \cdot \dot{\alpha}(\mathrm{Baexp}[\![A]\!](\dot{\gamma}(r))\gamma(p)) \\
= & \quad \wr \text{def. (29) of } \mathrm{Baexp}[\![A]\!] \int \\
 & \lambda r \in \mathbb{V} \mapsto L \cdot \lambda p \in L \cdot \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash A \Mapsto i\}) \ .
\end{array}
$$

If $r$ is the infimum $\lambda Y \cdot \bot$ where the infimum $\bot$ of $L$ is such that $\gamma(\bot) = \emptyset$, then $\dot{\gamma}(r) = \emptyset$ whence

$$
\begin{array}{ll}
 & \alpha^{\triangleleft}(\mathrm{Baexp}[\![A]\!])(\lambda Y \cdot \bot)p \\
= & \quad \wr \text{def. (19) of } \dot{\gamma} \int \\
 & \dot{\alpha}(\emptyset) \\
= & \quad \wr \text{def. (18) of } \dot{\alpha} \int \\
 & \lambda Y \cdot \bot \ .
\end{array}
$$

Given any $r \in \mathbb{V} \mapsto L$, $r \neq \lambda Y \cdot \bot$ or $\gamma(\bot) \neq \emptyset$ and $p \in L$, we proceed by structural induction on the arithmetic expression $A$.

1 — When $A = \mathsf{n} \in \mathrm{Nat}$ is a number, we have

$$
\begin{array}{ll}
 & \alpha^{\triangleleft}(\mathrm{Baexp}[\![\mathsf{n}]\!])(r)p \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash \mathsf{n} \Mapsto i\}) \\
= & \quad \wr \text{def. (23) of } \rho \vdash \mathsf{n} \Mapsto i \int \\
 & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \underline{\mathsf{n}} \in \gamma(p) \cap \mathbb{I}\}) \\
= & \quad \wr \text{def. conditional } (\ldots \ \textbf{?} \ \ldots \ \textbf{¿} \ \ldots) \int \\
 & (\underline{\mathsf{n}} \in \gamma(p) \cap \mathbb{I} \ \textbf{?} \ \dot{\alpha}(\dot{\gamma}(r)) \ \textbf{¿} \ \dot{\alpha}(\emptyset)) \\
\dot{\sqsubseteq} & \quad \wr \dot{\alpha} \circ \dot{\gamma} \text{ is reductive (7) and def. (18) of } \dot{\alpha} \int \\
 & (\underline{\mathsf{n}} \in \gamma(p) \cap \mathbb{I} \ \textbf{?} \ r \ \textbf{¿} \ \lambda Y \cdot \bot) \\
= & \quad \wr \text{by defining } \mathsf{n}^{\triangleleft}(p) \triangleq (\underline{\mathsf{n}} \in \gamma(p) \cap \mathbb{I}) \int \\
 & (\mathsf{n}^{\triangleleft}(p) \ \textbf{?} \ r \ \textbf{¿} \ \lambda Y \cdot \bot) \\
= & \quad \wr \text{by defining } \mathrm{Baexp}^{\triangleleft}[\![\mathsf{n}]\!](r)p \triangleq (\mathsf{n}^{\triangleleft}(p) \ \textbf{?} \ r \ \textbf{¿} \ \lambda Y \cdot \bot) \int \\
 & \mathrm{Baexp}^{\triangleleft}[\![\mathsf{n}]\!](r)p \ .
\end{array}
$$

2 — When $A = \mathsf{X} \in \mathbb{V}$ is a variable, we have

$$
\begin{array}{ll}
& \alpha^{\triangleleft}(\mathrm{Baexp}[\![\mathsf{X}]\!])(r)p \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash \mathsf{X} \Rrightarrow i\}) \\
= & \quad \wr \text{def. (24) of } \rho \vdash \mathsf{X} \Rrightarrow i \wr \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho(\mathsf{X}) \in \gamma(p) \cap \mathbb{I}\}) \\
\dot{\sqsubseteq} & \quad \wr [\gamma \circ \alpha \text{ is extensive (6) and } \dot{\alpha} \text{ is monotone (5)} \wr \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho(\mathsf{X}) \in \gamma(p) \cap \gamma \circ \alpha(\mathbb{I})\}) \\
= & \quad \wr \text{def. (19) of } \dot{\gamma} \wr \\
& \dot{\alpha}(\{\rho \mid \forall \mathsf{Y} \neq \mathsf{X} : \rho(\mathsf{Y}) \in \gamma(r(\mathsf{Y})) \wedge \rho(\mathsf{X}) \in \gamma(r(\mathsf{X})) \cap \gamma(p) \cap \gamma \circ \alpha(\mathbb{I})\}) \\
= & \quad \wr \gamma \text{ is a complete meet morphism} \wr \\
& \dot{\alpha}(\{\rho \mid \forall \mathsf{Y} \neq \mathsf{X} : \rho(\mathsf{Y}) \in \gamma(r(\mathsf{Y})) \wedge \rho(\mathsf{X}) \in \gamma(r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I}))\}) \\
= & \quad \wr \text{def. (16) of environment assignment} \wr \\
& \dot{\alpha}(\{\rho \mid \forall \mathsf{Y} \neq \mathsf{X} : \rho(\mathsf{Y}) \in \gamma(r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I})](\mathsf{Y})) \wedge \rho(\mathsf{X}) \in \gamma(r[\mathsf{X} \leftarrow \\
& r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I})](\mathsf{X})\}) \\
= & \quad \wr \text{def. (19) of } \dot{\gamma} \wr \\
& \dot{\alpha}(\{\rho \mid \rho \in \dot{\gamma}(r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I})])\} \\
= & \quad \wr \text{set notation} \wr \\
& \dot{\alpha}(\dot{\gamma}(r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I})])) \\
\dot{\sqsubseteq} & \quad \wr \dot{\alpha} \circ \dot{\gamma} \text{ is reductive (7)} \wr \\
& r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap \alpha(\mathbb{I})] \\
\dot{\sqsubseteq} & \quad \wr \text{def. (36) of } ?^{\triangleright} \wr \\
& r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap ?^{\triangleright}] \\
= & \quad \wr \text{by defining } \mathrm{Baexp}^{\triangleleft}[\![\mathsf{X}]\!](r)p \triangleq r[\mathsf{X} \leftarrow r(\mathsf{X}) \sqcap p \sqcap ?^{\triangleright}] \wr \\
& \mathrm{Baexp}^{\triangleleft}[\![\mathsf{X}]\!](r)p\ .
\end{array}
$$

3 — When $A = ?$ is random, we have

$$
\begin{array}{ll}
& \alpha^{\triangleleft}(\mathrm{Baexp}[\![?]\!])(r)p \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash ? \Rrightarrow i\}) \\
= & \quad \wr \text{def. (25) of } \rho \vdash ? \Rrightarrow i \wr \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \gamma(p) \cap \mathbb{I} \neq \emptyset\}) \\
= & \quad \wr \text{def. conditional } (\ldots \mathbin{?} \ldots ¿ \ldots) \wr \\
& (\gamma(p) \cap \mathbb{I} = \emptyset \mathbin{?} \dot{\alpha}(\emptyset) ¿ \dot{\alpha}(\dot{\gamma}(r))) \\
\dot{\sqsubseteq} & \quad \wr \text{def. (18) of } \dot{\alpha} \text{ and } \dot{\alpha} \circ \dot{\gamma} \text{ reductive (7)} \wr \\
& (\gamma(p) \cap \mathbb{I} = \emptyset \mathbin{?} \lambda \mathsf{Y} \bullet \bot ¿ r) \\
= & \quad \wr \text{negation} \wr \\
& (\gamma(p) \cap \mathbb{I} \neq \emptyset \mathbin{?} r ¿ \lambda \mathsf{Y} \bullet \bot) \\
= & \quad \wr \text{by defining } ?^{\triangleleft}(p) \triangleq (\gamma(p) \cap \mathbb{I} \neq \emptyset) \wr \\
& (?^{\triangleleft}(p) \mathbin{?} r ¿ \lambda \mathsf{Y} \bullet \bot) \\
= & \quad \wr \text{by defining } \mathrm{Baexp}^{\triangleleft}[\![?]\!] \triangleq (?^{\triangleleft}(p) \mathbin{?} r ¿ \lambda \mathsf{Y} \bullet \bot) \wr \\
& \mathrm{Baexp}^{\triangleleft}[\![?]\!](r)p\ .
\end{array}
$$

4 — When $A = \mathsf{u}\ A'$ is a unary operation, we have

$$
\begin{array}{ll}
& \alpha^{\triangleleft}(\mathrm{Baexp}[\![\mathsf{u}\ A']\!])(r)p \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash \mathsf{u}\ A' \Rrightarrow i\}) \\
= & \quad \wr \text{def. (4) of } \rho \vdash \mathsf{u}\ A' \Rrightarrow i \wr
\end{array}
$$

$$
\begin{aligned}
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' : \rho \vdash A' \Mapsto i' \wedge \underline{\mathsf{u}}\, i' \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{set theory} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A' \Mapsto v\} : \rho \vdash A' \Mapsto i' \wedge \underline{\mathsf{u}}\, i' \in \gamma(p) \cap \mathbb{I}\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \gamma \circ \alpha \text{ extensive (6) and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(\alpha(\{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A' \Mapsto v\})) : \rho \vdash A' \Mapsto i' \wedge \underline{\mathsf{u}}\, i' \in \gamma(p) \cap \mathbb{I}\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \text{(33) implying } \mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r \sqsupseteq \alpha(\{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A' \Mapsto v\}), \\
& \quad \gamma \text{ and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r) : \rho \vdash A' \Mapsto i' \wedge \underline{\mathsf{u}}\, i' \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{def. (21) of } \underline{\mathsf{u}} \text{ (such that } \underline{\mathsf{u}}\, i' \in \mathbb{I} \text{ only if } i' \in \mathbb{I}) \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r) \cap \mathbb{I} : \rho \vdash A' \Mapsto i' \wedge \underline{\mathsf{u}}\, i' \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{set theory} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \{i \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r) \mid \underline{\mathsf{u}}\, i \in \gamma(p) \cap \mathbb{I}\} \cap \mathbb{I} : \rho \vdash A' \Mapsto i'\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \gamma \circ \alpha \text{ extensive (6) and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(\alpha(\{i \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r) \mid \underline{\mathsf{u}}\, i \in \gamma(p) \cap \mathbb{I}\})) \cap \mathbb{I} : \rho \vdash A' \Mapsto i'\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \text{defining } \mathsf{u}^{\triangleleft} \text{ such that } \mathsf{u}^{\triangleleft}(q, p) \sqsupseteq \alpha(\{i \in \gamma(q) \mid \underline{\mathsf{u}}\, i \in \gamma(p) \cap \mathbb{I}\}), \\
& \quad \gamma \text{ and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(\mathsf{u}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r, p)) \cap \mathbb{I} : \rho \vdash A' \Mapsto i'\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \text{induction hypothesis (38) implying } \mathrm{Baexp}^{\triangleleft}\llbracket A' \rrbracket (r) p \mathrel{\dot\sqsupseteq} \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i' \in \gamma(p) \cap \mathbb{I} : \\
& \quad \rho \vdash A' \Mapsto i'\}) \wr \\
& \mathrm{Baexp}^{\triangleleft}\llbracket A' \rrbracket (r)(\mathsf{u}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r, p)) \\
= \quad & \wr \text{defining } \mathrm{Baexp}^{\triangleleft}\llbracket \mathsf{u}\, A' \rrbracket (r) p \triangleq \mathrm{Baexp}^{\triangleleft}\llbracket A' \rrbracket (r)(\mathsf{u}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}\llbracket A' \rrbracket r, p)) \wr \\
& \mathrm{Baexp}^{\triangleleft}\llbracket \mathsf{u}\, A' \rrbracket (r) p \;.
\end{aligned}
$$

5 — When $A = A_1 \mathsf{b} A_2$ is a binary operation, we have

$$
\begin{aligned}
& \alpha^{\triangleleft}(\mathrm{Baexp}\llbracket A_1 \mathsf{b} A_2 \rrbracket)(r) p \\
= \quad & \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i \in \gamma(p) \cap \mathbb{I} : \rho \vdash A_1 \mathsf{b} A_2 \Mapsto i\}) \\
= \quad & \wr \text{def. (27) of } \rho \vdash A_1 \mathsf{b} A_2 \Mapsto i \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1, i_2 : \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2 \wedge i_1 \,\underline{\mathsf{b}}\, i_2 \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{set theory} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A_1 \Mapsto v\} : \\
& \qquad \exists i_2 \in \{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A_2 \Mapsto v\} : \\
& \qquad\quad \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2 \wedge i_1 \,\underline{\mathsf{b}}\, i_2 \in \gamma(p) \cap \mathbb{I}\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \gamma \circ \alpha \text{ extensive (6) and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \gamma(\alpha(\{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A_1 \Mapsto v\})) : \\
& \qquad \exists i_2 \in \gamma(\alpha(\{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash A_2 \Mapsto v\})) : \\
& \qquad\quad \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2 \wedge i_1 \,\underline{\mathsf{b}}\, i_2 \in \gamma(p) \cap \mathbb{I}\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \text{(33) implying } \mathrm{Faexp}^{\triangleright}\llbracket A_i \rrbracket r \sqsupseteq \alpha(\{v \mid \exists \rho' \in \dot\gamma(r) : \rho' \vdash_i \Mapsto v\}), i = 1, 2, \\
& \quad \gamma \text{ and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_1 \rrbracket r) : \exists i_2 \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_2 \rrbracket r) : \\
& \qquad \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2 \wedge i_1 \,\underline{\mathsf{b}}\, i_2 \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{def. (22) of } \underline{\mathsf{b}} \text{ (such that } i_1 \,\underline{\mathsf{b}}\, i_2 \in \mathbb{I} \text{ only if } i_1, i_2 \in \mathbb{I}) \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_1 \rrbracket r) \cap \mathbb{I} : \exists i_2 \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_2 \rrbracket r) \cap \mathbb{I} : \\
& \qquad \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2 \wedge i_1 \,\underline{\mathsf{b}}\, i_2 \in \gamma(p) \cap \mathbb{I}\}) \\
= \quad & \wr \text{set theory} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists \langle i_1,\ i_2 \rangle \in \{\langle i_1',\ i_2' \rangle \in \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_1 \rrbracket r) \times \gamma(\mathrm{Faexp}^{\triangleright}\llbracket A_2 \rrbracket r) \mid \\
& \qquad i_1' \,\underline{\mathsf{b}}\, i_2' \in \gamma(p) \cap \mathbb{I}\} \cap (\mathbb{I} \times \mathbb{I}) : \rho \vdash A_1 \Mapsto i_1 \wedge \rho \vdash A_2 \Mapsto i_2\}) \\
\mathrel{\dot\sqsubseteq} \quad & \wr \gamma^2 \circ \alpha^2 \text{ extensive (13), (6) and } \dot\alpha \text{ monotone (20), (5)} \wr
\end{aligned}
$$

$$
\begin{array}{ll}
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\alpha^2(\{\langle i'_1,\ i'_2\rangle \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r) \times \gamma(\mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \mid \\
& \qquad\qquad i'_1 \;\underline{\mathsf{b}}\; i'_2 \in \gamma(p) \cap \mathbb{I}\})) \cap (\mathbb{I} \times \mathbb{I}) : \rho \vdash A_1 \Rightarrow i_1 \wedge \rho \vdash A_2 \Rightarrow i_2\}) \\
\dot{\sqsubseteq} & \quad \wr \text{defining } \mathsf{b}^{\triangleleft} \text{ such that} \\
& \qquad \mathsf{b}^{\triangleleft}(q_1, q_2, p) \sqsupseteq^2 \alpha^2(\{\langle i'_1,\ i'_2\rangle \in \gamma^2(\langle q_1,\ q_2\rangle) \mid i'_1 \;\underline{\mathsf{b}}\; i'_2 \in \gamma(p) \cap \mathbb{I}\}), \\
& \qquad \gamma^2 \text{ and } \dot{\alpha} \text{ monotone (20), (5)} \wr \\
& \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)) \cap (\mathbb{I} \times \mathbb{I}) : \\
& \qquad\qquad \rho \vdash A_1 \Rightarrow i_1 \wedge \rho \vdash A_2 \Rightarrow i_2\}) \\
= & \quad \wr \text{let notation} \wr \\
& \mathrm{let}\ \langle p_1,\ p_2\rangle = \mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)\ \mathrm{in} \\
& \quad \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\langle p_1,\ p_2\rangle) \cap (\mathbb{I} \times \mathbb{I}) : \rho \vdash A_1 \Rightarrow i_1 \wedge \rho \vdash A_2 \Rightarrow i_2\}) \\
= & \quad \wr \text{def. (12) of } \gamma^2 \text{ and } \dot{\alpha} \text{ monotone (20), (5)} \wr \\
& \mathrm{let}\ \langle p_1,\ p_2\rangle = \mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)\ \mathrm{in} \\
& \quad \dot{\alpha}(\{\rho_1 \in \dot{\gamma}(r) \mid \exists i_1 \in \gamma p_1 \cap \mathbb{I} : \rho_1 \vdash A_1 \Rightarrow i_1\} \cap \\
& \qquad \{\rho_2 \in \dot{\gamma}(r) \mid \exists i_2 \in \gamma p_2 \cap \mathbb{I} : \rho_2 \vdash A_2 \Rightarrow i_2\}) \\
= & \quad \wr \dot{\alpha} \text{ complete join morphism} \wr \\
& \mathrm{let}\ \langle p_1,\ p_2\rangle = \mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)\ \mathrm{in} \\
& \quad \dot{\alpha}(\{\rho_1 \in \dot{\gamma}(r) \mid \exists i_1 \in \gamma p_1 \cap \mathbb{I} : \rho_1 \vdash A_1 \Rightarrow i_1\}) \\
& \qquad \dot{\sqcap}\, \dot{\alpha}(\{\rho_2 \in \dot{\gamma}(r) \mid \exists i_2 \in \gamma p_2 \cap \mathbb{I} : \rho_2 \vdash A_2 \Rightarrow i_2\}) \\
\dot{\sqsubseteq} & \quad \wr \text{induction hypothesis (38) implying} \\
& \qquad \mathrm{Baexp}^{\triangleleft}[\![A']\!](r)p \;\dot{\sqsupseteq}\; \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i' \in \gamma(p) \cap \mathbb{I} : \rho \vdash A' \Rightarrow i'\}) \wr \\
& \mathrm{let}\ \langle p_1,\ p_2\rangle = \mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)\ \mathrm{in} \\
& \quad \mathrm{Baexp}^{\triangleleft}[\![A_1]\!](r)p_1 \;\dot{\sqcap}\; \mathrm{Baexp}^{\triangleleft}[\![A_2]\!](r)p_2 \\
= & \quad \wr \text{defining } \mathrm{Baexp}^{\triangleleft}[\![A_1 \;\mathsf{b}\; A_2]\!](r)p \triangleq \\
& \qquad\quad \mathrm{let}\ \langle p_1,\ p_2\rangle = \mathsf{b}^{\triangleleft}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r, p)\ \mathrm{in} \\
& \qquad\qquad \mathrm{Baexp}^{\triangleleft}[\![A_1]\!](r)p_1 \;\dot{\sqcap}\; \mathrm{Baexp}^{\triangleleft}[\![A_2]\!](r)p_2 \wr \\
& \mathrm{Baexp}^{\triangleleft}[\![A_1 \;\mathsf{b}\; A_2]\!](r)p\ .
\end{array}
$$

In conclusion, we have designed the backward abstract interpretation $\mathrm{Baexp}^{\triangleleft}$ of arithmetic expressions in such a way that it satisfies the soundness requirement (38) as summarized in Fig. 7.

For all $p \in L$ and by induction on $A$, the operator $\lambda r \bullet \mathrm{Baexp}^{\triangleleft}[\![A]\!](r)p$ on $\mathbb{V} \mapsto L$ is $\dot{\sqsubseteq}$-reductive and monotonic.

### 8.6 Generic backward/bottom-up static analyzer of arithmetic expressions

A rapid prototyping of Fig. 7 with signature

```
module type Baexp_signature =
 functor (L: Abstract_Lattice_Algebra_signature) ->
 functor (E: Abstract_Env_Algebra_signature) ->
 functor (Faexp: Faexp_signature) ->
 sig
  open Abstract_Syntax
  (* generic backward abstract interpretation of arithmetic operations *)
  val baexp : aexp ->  E (L).env -> L.lat -> E (L).env
 end;;
```

is given by the following implementation

```
module Baexp_implementation =
 functor (L: Abstract_Lattice_Algebra_signature) ->
 functor (E: Abstract_Env_Algebra_signature) ->
```

$$
\begin{array}{rcll}
\mathrm{Baexp}^{\lhd}[\![A]\!](\lambda \mathsf{Y}\bullet\bot)\,p & \stackrel{\Delta}{=} & \lambda\mathsf{Y}\bullet\bot \quad \text{if} \quad \gamma(\bot)=\emptyset & (39)\\
\mathrm{Baexp}^{\lhd}[\![\mathsf{n}]\!](r)\,p & \stackrel{\Delta}{=} & (\mathsf{n}^{\lhd}(p)\ \mathbf{?}\ r\ \mathbf{¿}\ \lambda\mathsf{Y}\bullet\bot) & \\
\mathrm{Baexp}^{\lhd}[\![\mathsf{X}]\!](r)\,p & \stackrel{\Delta}{=} & r[\mathsf{X}\leftarrow r(\mathsf{X})\sqcap p\sqcap ?^{\rhd}] & (40)\\
\mathrm{Baexp}^{\lhd}[\![?]\!](r)\,p & \stackrel{\Delta}{=} & (?^{\lhd}(p)\ \mathbf{?}\ r\ \mathbf{¿}\ \lambda\mathsf{Y}\bullet\bot) & \\
\mathrm{Baexp}^{\lhd}[\![\mathsf{u}\ A']\!](r)\,p & \stackrel{\Delta}{=} & \mathrm{Baexp}^{\lhd}[\![A']\!](r)(\mathsf{u}^{\lhd}(\mathrm{Faexp}^{\rhd}[\![A']\!]r,\ p)) & \\
\mathrm{Baexp}^{\lhd}[\![A_1\ \mathsf{b}\ A_2]\!](r)\,p & \stackrel{\Delta}{=} & \text{let } \langle p_1,\ p_2\rangle = \mathsf{b}^{\lhd}(\mathrm{Faexp}^{\rhd}[\![A_1]\!]r,\ \mathrm{Faexp}^{\rhd}[\![A_2]\!]r,\ p) \text{ in} & \\
 & & \quad \mathrm{Baexp}^{\lhd}[\![A_1]\!](r)\,p_1 \mathbin{\dot\sqcap} \mathrm{Baexp}^{\lhd}[\![A_2]\!](r)\,p_2 &
\end{array}
$$

parameterized by the following backward abstract operations on $L$

$$
\begin{array}{rcll}
\mathsf{n}^{\lhd}(p) & \stackrel{\Delta}{=} & (\underline{\mathsf{n}}\in\gamma(p)\cap\mathbb{I}) & (41)\\
?^{\lhd}(p) & \stackrel{\Delta}{=} & (\gamma(p)\cap\mathbb{I}\neq\emptyset) & (42)\\
\mathsf{u}^{\lhd}(q,\ p) & \sqsupseteq & \alpha(\{i\in\gamma(q)\mid \underline{\mathsf{u}}\,i\in\gamma(p)\cap\mathbb{I}\}) & (43)\\
\mathsf{b}^{\lhd}(q_1, q_2,\ p) & \sqsupseteq^2 & \alpha^2(\{\langle i_1,\ i_2\rangle\in\gamma^2(\langle q_1,\ q_2\rangle)\mid i_1\ \underline{\mathsf{b}}\ i_2\in\gamma(p)\cap\mathbb{I}\}) & (44)
\end{array}
$$

Figure 7: Backward abstract interpretation of arithmetic expressions

```
 functor (Faexp: Faexp_signature) ->
struct
 open Abstract_Syntax
 (* generic abstract environments                                    *)
 module E' = E (L)
 (* generic forward abstract interpretation of arithmetic operations  *)
 module Faexp' = Faexp(L)(E)
 (* generic backward abstract interpretation of arithmetic operations *)
 let rec baexp' a r p =
   match a with
   | (INT i) -> if (L.b_INT i p) then r else (E'.bot ())
   | (VAR v) ->
       (E'.set r v (L.meet (L.meet (E'.get r v) p) (L.f_RANDOM ())))
   | RANDOM -> if (L.b_RANDOM p) then r else (E'.bot ())
   | (UMINUS a1) -> (baexp' a1 r (L.b_UMINUS (Faexp'.faexp a1 r) p))
   | (UPLUS a1)  -> (baexp' a1 r (L.b_UPLUS (Faexp'.faexp a1 r) p))
   | (PLUS (a1, a2)) ->
     let (p1,p2) = (L.b_PLUS (Faexp'.faexp a1 r) (Faexp'.faexp a2 r) p)
       in (E'.meet (baexp' a1 r p1) (baexp' a2 r p2))
   | (MINUS (a1, a2))  ->
     let (p1,p2) = (L.b_MINUS (Faexp'.faexp a1 r) (Faexp'.faexp a2 r) p)
       in (E'.meet (baexp' a1 r p1) (baexp' a2 r p2))
   | (TIMES (a1, a2))  ->
     let (p1,p2) = (L.b_TIMES (Faexp'.faexp a1 r) (Faexp'.faexp a2 r) p)
       in (E'.meet (baexp' a1 r p1) (baexp' a2 r p2))
   | (DIV (a1, a2))  ->
     let (p1,p2) = (L.b_DIV (Faexp'.faexp a1 r) (Faexp'.faexp a2 r) p)
       in (E'.meet (baexp' a1 r p1) (baexp' a2 r p2))
   | (MOD (a1, a2))  ->
     let (p1,p2) = (L.b_MOD (Faexp'.faexp a1 r) (Faexp'.faexp a2 r) p)
       in (E'.meet (baexp' a1 r p1) (baexp' a2 r p2))
 let baexp a r p =
```

```
  if (E'.is_bot r) & (L.isbotempty ()) then (E'.bot ()) else baexp' a r p
end;;

module Baexp = (Baexp_implementation:Baexp_signature);;
```

The operations on abstract value properties which are used for the backward abstract interpretation of arithmetic expressions of Fig. 7 must be provided with the module implementing each particular algebra of abstract properties, as follows

```
module type Abstract_Lattice_Algebra_signature =
  sig
    (* complete lattice of abstract properties of values          *)
    type lat                                (* abstract properties  *)
    ...
    (* forward abstract interpretation of arithmetic expressions  *)
    ...
    (* backward abstract interpretation of arithmetic expressions *)
    val b_INT    : string -> lat -> bool
    val b_RANDOM : lat -> bool
    val b_UMINUS : lat -> lat -> lat
    val b_UPLUS  : lat -> lat -> lat
    val b_PLUS   : lat -> lat -> lat -> lat * lat
    val b_MINUS  : lat -> lat -> lat -> lat * lat
    val b_TIMES  : lat -> lat -> lat -> lat * lat
    val b_DIV    : lat -> lat -> lat -> lat * lat
    val b_MOD    : lat -> lat -> lat -> lat * lat
    ...
  end;;
```

The next section is an example of calculational design of such abstract operations for the initialization and simple sign analysis.

### 8.7 *Initialization and simple sign abstract backward arithmetic operations*

In the abstract interpretation (40) of variables, we have

$$?^{\triangleright} \;=\; \mathtt{INI}$$

by definition (15) of $\alpha$. From the definition (41) of $\mathsf{n}^{\triangleleft}$ and (14) of $\gamma$, we directly get by case analysis

| $\mathsf{n}^{\triangleleft}(p)$ \ $p$ | BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|
| $\underline{\mathsf{n}} \in [\mathtt{min\_int}, -1]$ | ff | tt | ff | ff | tt | ff | tt |
| $\underline{\mathsf{n}} = 0$ | ff | ff | tt | ff | tt | ff | tt |
| $\underline{\mathsf{n}} \in [1, \mathtt{max\_int}]$ | ff | ff | ff | tt | tt | ff | tt |
| $\underline{\mathsf{n}} < \mathtt{min\_int} \vee \underline{\mathsf{n}} > \mathtt{max\_int}$ | ff | ff | ff | ff | ff | ff | ff |

From the definition (42) of $?^{\triangleleft}$ and (14) of $\gamma$, we directly get by case analysis

| $p$ | BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|
| $?^{\triangleleft}(p)$ | ff | tt | tt | tt | tt | ff | tt |

For the backward unary arithmetic operations (43), we have

| $p$ | BOT | NEG | ZERO | POS | INI | ERR | TOP |
|---|---|---|---|---|---|---|---|
| $+^{\triangleleft}(q, p)$ | BOT | $q \sqcap \mathtt{NEG}$ | $q \sqcap \mathtt{ZERO}$ | $q \sqcap \mathtt{POS}$ | $q \sqcap \mathtt{INI}$ | BOT | $q \sqcap \mathtt{INI}$ |
| $-^{\triangleleft}(q, p)$ | BOT | $q \sqcap \mathtt{POS}$ | $q \sqcap \mathtt{ZERO}$ | $q \sqcap \mathtt{NEG}$ | $q \sqcap \mathtt{INI}$ | BOT | $q \sqcap \mathtt{INI}$ |

Let us consider a few typical cases.

1 —— If $p = \texttt{BOT}$ or $p = \texttt{ERR}$ then by (14),

$$\underline{\mathsf{u}}\, i \in \gamma(p) \cap \mathbb{I} \subseteq \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\} \cap [\texttt{min\_int}, \texttt{max\_int}] = \emptyset$$

is false so that $\mathsf{u}^{\triangleleft}(q, p) = \alpha(\emptyset) = \texttt{BOT}$.

2 —— If $p = \texttt{POS}$ then by (14), $\underline{-}\, i \in \gamma(p) \cap \mathbb{I} = [1, \texttt{max\_int}]$ if and only if $\wr$by def. (21) of $\underline{-} \int i \in [\texttt{min\_int} + 1, -1]$ so that $-^{\triangleleft}(q, p) = \alpha(\gamma(q) \cap [\texttt{min\_int} + 1, -1]) \subseteq \alpha(\gamma(q) \cap \gamma(\texttt{NEG}))$ by (14). But $\gamma$ preserves meets whence this is equal to $\alpha(\gamma(q \sqcap \texttt{NEG})) \sqsubseteq q \sqcap \texttt{NEG}$ since $\alpha \circ \gamma$ is reductive (7).

3 —— If $p = \texttt{INI}$ or $p = \texttt{TOP}$ then by (14), $\underline{-}\, i \in \gamma(p) \cap \mathbb{I} = [\texttt{min\_int}, \texttt{max\_int}]$ if and only if $\wr$by def. (21) of $\underline{-} \int i \in [\texttt{min\_int} + 1, \texttt{max\_int}]$ so that $-^{\triangleleft}(q, p) = \alpha(\gamma(q) \cap [\texttt{min\_int} + 1, \texttt{max\_int}]) \subseteq \alpha(\gamma(q) \cap \gamma(\texttt{INI}))$ by (14). But $\gamma$ preserves meets whence this is equal to $\alpha(\gamma(q \sqcap \texttt{INI})) \sqsubseteq q \sqcap \texttt{INI}$ since $\alpha \circ \gamma$ is reductive (7).

For the backward binary arithmetic operations (44), we have

$$\begin{aligned}
/^{\triangleleft}(q_1, q_2, p) \;&\triangleq\; \texttt{mod}^{\triangleleft}(q_1, q_2, p) \;\triangleq\\
&(q_1 \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ERR}\} \vee q_2 \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ZERO}, \texttt{ERR}\} \vee p \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ERR}\} \;\textbf{?}\; \langle \texttt{BOT},\ \texttt{BOT}\rangle\\
&\quad \textbf{¿}\; (p = \texttt{POS} \;\textbf{?}\; \mathit{smash}(\langle q_1 \sqcap \texttt{POS},\ q_2 \sqcap \texttt{POS}\rangle) \;\textbf{¿}\; \langle q_1 \sqcap \texttt{INI},\ q_2 \sqcap \texttt{POS}\rangle))\\
\mathit{smash}(\langle x,\ y\rangle) \;&\triangleq\; (x = \texttt{BOT} \vee y = \texttt{BOT} \;\textbf{?}\; \langle \texttt{BOT},\ \texttt{BOT}\rangle \;\textbf{¿}\; \langle x,\ y\rangle)\ .
\end{aligned}$$

If $\mathsf{b} \in \{/, \texttt{mod}\}$ and $q_1 \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ERR}\}$ or $q_2 \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ZERO}, \texttt{ERR}\}$ then $i_1 \in \gamma(q_1) \subseteq [\texttt{min\_int}, -1] \cup \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}$ or $i_2 \in \gamma(q_2) \subseteq [\texttt{min\_int}, 0] \cup \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}$ in which case $i_1 \,\underline{\mathsf{b}}\, i_2 \notin \mathbb{I}$ by (22). If follows that $\mathsf{b}^{\triangleleft}(q_1, q_2, p) = \alpha^2(\emptyset) = \langle \texttt{BOT},\ \texttt{BOT}\rangle$ by (11) and (15).

If $p \in \{\texttt{BOT}, \texttt{NEG}, \texttt{ERR}\}$ then $i_1 \,\underline{\mathsf{b}}\, i_2 \notin \gamma(p) \cap \mathbb{I} \subseteq [\texttt{min\_int}, -1]$ in contradiction with (22) showing that $i_1 \,\underline{\mathsf{b}}\, i_2$ is not negative. Again $\mathsf{b}^{\triangleleft}(q_1, q_2, p) = \alpha^2(\emptyset) = \langle \texttt{BOT},\ \texttt{BOT}\rangle$ by (11) and (15).

Otherwise to have $i_1 \,\underline{\mathsf{b}}\, i_2 \in \mathbb{I}$, we must have $i_1 \in [0, \texttt{max\_int}]$ and $i_2 \in [1, \texttt{max\_int}]$ whence necessarily $i_1 \in \gamma(\texttt{INI})$ and $i_2 \in \gamma(\texttt{POS})$ so that $\alpha^2(\gamma^2(\langle q_1 \sqcap \texttt{INI},\ q_2 \sqcap \texttt{POS}\rangle)) \sqsubseteq^2 \langle q_1 \sqcap \texttt{INI},\ q_2 \sqcap \texttt{POS}\rangle \triangleq \mathsf{b}^{\triangleleft}(q_1, q_2, p)$. Moreover the quotient is strictly positive only if the dividend is non zero.

With the same reasoning, for addition $+^{\triangleleft}$, we have

$$\begin{aligned}
+^{\triangleleft}(q_1, q_2, p) &= \langle \texttt{BOT},\ \texttt{BOT}\rangle && \text{if} \quad q_1 \in \{\texttt{BOT}, \texttt{ERR}\} \vee q_2 \in \{\texttt{BOT}, \texttt{ERR}\} \vee\\
& && \qquad p \in \{\texttt{BOT}, \texttt{ERR}\}\\
+^{\triangleleft}(q_1, q_2, p) &= \langle q_1 \sqcap \texttt{INI},\ q_2 \sqcap \texttt{INI}\rangle && \text{if} \quad p \in \{\texttt{INI}, \texttt{TOP}\}\ .
\end{aligned}$$

Otherwise

| $+^{\triangleleft}(q_1, q_2, \texttt{NEG})$ | | $q_2$: NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨NEG, NEG⟩ | ⟨NEG, ZERO⟩ | ⟨NEG, POS⟩ | ⟨NEG, INI⟩ |
| | ZERO | ⟨ZERO, NEG⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨ZERO, NEG⟩ |
| | POS | ⟨POS, NEG⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨POS, NEG⟩ |
| | INI, TOP | ⟨INI, NEG⟩ | ⟨NEG, ZERO⟩ | ⟨NEG, POS⟩ | ⟨INI, INI⟩ |

| $+^{\triangleleft}(q_1, q_2, \texttt{ZERO})$ | | $q_2$: NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨NEG, POS⟩ | ⟨NEG, POS⟩ |
| | ZERO | ⟨BOT, BOT⟩ | ⟨ZERO, ZERO⟩ | ⟨BOT, BOT⟩ | ⟨ZERO, ZERO⟩ |
| | POS | ⟨POS, NEG⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨POS, NEG⟩ |
| | INI, TOP | ⟨POS, NEG⟩ | ⟨ZERO, ZERO⟩ | ⟨NEG, POS⟩ | ⟨INI, INI⟩ |

| $+^{\triangleleft}(q_1, q_2, \texttt{POS})$ | | $q_2$ NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨NEG, POS⟩ | ⟨NEG, POS⟩ |
| | ZERO | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨ZERO, POS⟩ | ⟨ZERO, POS⟩ |
| | POS | ⟨POS, NEG⟩ | ⟨POS, ZERO⟩ | ⟨POS, POS⟩ | ⟨POS, INI⟩ |
| | INI, TOP | ⟨POS, NEG⟩ | ⟨POS, ZERO⟩ | ⟨INI, POS⟩ | ⟨INI, INI⟩ |

The backward ternary substraction operation $-^{\triangleleft}$ is defined as

$$\begin{aligned} -^{\triangleleft}(q_1, q_2, p) \quad &\triangleq \quad \text{let } (r_1, r_2) = -^{\triangleleft}(q_1, -^{\triangleright}(q_2), p) \text{ in} \\ &\qquad (r_1, -^{\triangleright}(r_2)) \ . \end{aligned}$$

The handling of the backward ternary multiplication operation $*^{\triangleleft}$ is similar

| $*^{\triangleleft}(q_1, q_2, \texttt{NEG})$ | | $q_2$ NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨NEG, POS⟩ | ⟨NEG, POS⟩ |
| | ZERO | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ |
| | POS | ⟨POS, NEG⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨POS, NEG⟩ |
| | INI, TOP | ⟨POS, NEG⟩ | ⟨BOT, BOT⟩ | ⟨NEG, POS⟩ | ⟨INI, INI⟩ |

| $*^{\triangleleft}(q_1, q_2, \texttt{ZERO})$ | | $q_2$ NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨BOT, BOT⟩ | ⟨NEG, ZERO⟩ | ⟨BOT, BOT⟩ | ⟨NEG, ZERO⟩ |
| | ZERO | ⟨ZERO, NEG⟩ | ⟨ZERO, ZERO⟩ | ⟨ZERO, POS⟩ | ⟨ZERO, INI⟩ |
| | POS | ⟨BOT, BOT⟩ | ⟨POS, ZERO⟩ | ⟨BOT, BOT⟩ | ⟨POS, ZERO⟩ |
| | INI, TOP | ⟨ZERO, NEG⟩ | ⟨INI, ZERO⟩ | ⟨ZERO, POS⟩ | ⟨INI, INI⟩ |

| $*^{\triangleleft}(q_1, q_2, \texttt{POS})$ | | $q_2$ NEG | ZERO | POS | INI, TOP |
|---|---|---|---|---|---|
| $q_1$ | NEG | ⟨NEG, NEG⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨NEG, NEG⟩ |
| | ZERO | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ |
| | POS | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨POS, POS⟩ | ⟨POS, POS⟩ |
| | INI, TOP | ⟨NEG, NEG⟩ | ⟨BOT, BOT⟩ | ⟨POS, POS⟩ | ⟨INI, INI⟩ |

## 9. Semantics of Boolean Expressions

### *9.1 Abstract syntax of boolean expressions*

We assume that boolean expressions are normalized according to the abstract syntax of Fig. 8. The normalization is specified by the following recursive rewriting rules

$$
\begin{array}{lrll}
\textit{Arithmetic expressions} & & & \\
\quad A_1, A_2 \in \text{Aexp} \ . & & & \\
\textit{Boolean expressions} & & & \\
\quad B, B_1, B_2 \in \text{Bexp} & ::= & \texttt{true} & \text{truth,} \\
& \mid & \texttt{false} & \text{falsity,} \\
& \mid & A_1 = A_2 \quad \mid \quad A_1 < A_2 & \text{arithmetic comparison,} \\
& \mid & B_1 \,\&\, B_2 & \text{conjunction,} \\
& \mid & B_1 \mid B_2 & \text{disjunction.}
\end{array}
$$

Figure 8: Abstract syntax of boolean expressions

$$
\begin{array}{rcl@{\qquad}rcl}
T(\texttt{true}) & \triangleq & \texttt{true}, & T(\neg\texttt{true}) & \triangleq & \texttt{false}, \\
T(\texttt{false}) & \triangleq & \texttt{false}, & T(\neg\texttt{false}) & \triangleq & \texttt{true}, \\
T(A_1 < A_2) & \triangleq & A_1 < A_2, & T(\neg(A_1 < A_2)) & \triangleq & T(A_1 >= A_2), \\
T(A_1 <= A_2) & \triangleq & (A_1 < A_2) \mid (A_1 = A_2), & T(\neg(A_1 <= A_2)) & \triangleq & T(A_1 > A_2), \\
T(A_1 = A_2) & \triangleq & A_1 = A_2, & T(\neg(A_1 = A_2)) & \triangleq & T(A_1 <> A_2), \\
T(A_1 <> A_2) & \triangleq & (A_1 < A_2) \mid (A_2 < A_1), & T(\neg(A_1 <> A_2)) & \triangleq & A_1 = A_2, \\
T(A_1 > A_2) & \triangleq & A_2 < A_1, & T(\neg(A_1 > A_2)) & \triangleq & T(A_1 <= A_2), \\
T(A_1 >= A_2) & \triangleq & (A_1 = A_2) \mid (A_2 < A_1), & T(\neg(A_1 >= A_2)) & \triangleq & A_1 < A_2, \\
T(B_1 \mid B_2) & \triangleq & T(B_1) \mid T(B_2) & T(\neg(B_1 \mid B_2)) & \triangleq & T(\neg(B_1)) \,\&\, T(\neg(B_2)), \\
T(B_1 \,\&\, B_2) & \triangleq & T(B_1) \,\&\, T(B_2), & T(\neg(B_1 \,\&\, B_2)) & \triangleq & T(\neg(B_1)) \mid T(\neg(B_2)), \\
& & & T(\neg(\neg(B))) & \triangleq & T(B).
\end{array}
$$

### *9.2 Machine booleans*

We let $\mathbb{B}$ be the logical boolean values and $\mathbb{B}_\Omega$ be the machine truth values (including errors $\mathbb{E} = \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}$)

$$
\mathbb{B} \triangleq \{\text{tt}, \text{ff}\}, \qquad \mathbb{B}_\Omega \triangleq \mathbb{B} \cup \mathbb{E} \ .
$$

We respectively write $\underline{\mathsf{c}} \in \mathbb{I}_\Omega \times \mathbb{I}_\Omega \mapsto \mathbb{B}_\Omega$ for the machine arithmetic comparison operation and $\mathsf{c} \in \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{B}$ for the mathematical arithmetic comparison operation corresponding to the language binary arithmetic comparison operators $\mathsf{c} \in \{<, <=, =, <>, >=, >\}$. Evaluation of operands, whence error propagation is left to right. We have $e \in \mathbb{E}$, $v \in \mathbb{I}_\Omega$, $i, i_1, i_2 \in \mathbb{I}$)

$$
\begin{aligned}
\Omega_e \ \underline{\mathsf{c}} \ v &\triangleq \Omega_e, \\
i \ \underline{\mathsf{c}} \ \Omega_e &\triangleq \Omega_e, \\
i_1 \ \underline{\mathsf{c}} \ i_2 &\triangleq i_1 \ \mathsf{c} \ i_2 \ .
\end{aligned}
\tag{45}
$$

We respectively write $\underline{\mathsf{u}} \in \mathbb{B}_\Omega \mapsto \mathbb{B}_\Omega$ for the machine boolean operation and $\mathsf{u} \in \mathbb{B} \mapsto \mathbb{B}$ for the mathematical boolean operation corresponding to the language unary operators $\mathsf{u} \in \{\neg\}$. Errors are propagated, so that we have ($e \in \mathbb{E}$, $b \in \mathbb{B}$)

$$
\begin{aligned}
\underline{\mathsf{u}}\,\Omega_e &\triangleq \Omega_e, \\
\underline{\mathsf{u}}\,b &\triangleq \mathsf{u}\,b \ .
\end{aligned}
$$

$$\rho \vdash \texttt{true} \Rrightarrow \text{tt}, \qquad \text{truth;} \tag{47}$$

$$\rho \vdash \texttt{false} \Rrightarrow \text{ff}, \qquad \text{falsity;} \tag{48}$$

$$\frac{\rho \vdash A_1 \Rrightarrow v_1,\ \rho \vdash A_2 \Rrightarrow v_2}{\rho \vdash A_1 \,\texttt{c}\, A_2 \Rrightarrow v_1 \,\underline{\texttt{c}}\, v_2}, \qquad \text{arithmetic comparisons;} \tag{49}$$

$$\frac{\rho \vdash B \Rrightarrow w}{\rho \vdash \texttt{u}\, B \Rrightarrow \underline{\texttt{u}}\, w}, \qquad \text{unary boolean operations;}$$

$$\frac{\rho \vdash B_1 \Rrightarrow w_1,\ \rho \vdash B_2 \Rrightarrow w_2}{\rho \vdash B_1 \,\texttt{b}\, B_2 \Rrightarrow w_1 \,\underline{\texttt{b}}\, w_2}, \qquad \text{binary boolean operations.}$$

Figure 9: Operational semantics of boolean expressions

We respectively write $\underline{\texttt{b}} \in \mathbb{B}_\Omega \times \mathbb{B}_\Omega \mapsto \mathbb{B}_\Omega$ for the machine boolean operation and $\texttt{b} \in \mathbb{B} \times \mathbb{B} \mapsto \mathbb{B}$ for the mathematical boolean operation corresponding to the language binary boolean operators $\texttt{b} \in \{\&, |\}$. Evaluation of operands, whence error propagation is left to right. We have ($e \in \mathbb{E}$, $w \in \mathbb{B}_\Omega$, $b, b_1, b_2 \in \mathbb{B}$)

$$\begin{aligned} \Omega_e \,\underline{\texttt{b}}\, w &\stackrel{\Delta}{=} \Omega_e, \\ b \,\underline{\texttt{b}}\, \Omega_e &\stackrel{\Delta}{=} \Omega_e, \\ b_1 \,\underline{\texttt{b}}\, b_2 &\stackrel{\Delta}{=} b_1 \,\texttt{b}\, i_2 \,. \end{aligned} \tag{46}$$

### 9.3 *Operational semantics of boolean expressions*

The big-step operational semantics [31] of boolean expressions involves judgements $\rho \vdash B \Rrightarrow b$ meaning that in environment $\rho$, the boolean expression $b$ may evaluate to $b \in \mathbb{B}_\Omega$. If is formally specified by the inference system of Fig. 9.

### 9.4 *Equivalence of boolean expressions*

In general, the semantics of a boolean expression $B$ is not the same as the semantics of its transformed form $T(B)$. This is because the rewriting rule $T(A_1 > A_2) = A_2 < A_1$ does not respect left to right evaluation whence the error propagation order. For example if $\rho(\texttt{X}) = \Omega_{\texttt{i}}$ then $\rho \vdash \texttt{X} > (\texttt{1} / \texttt{0}) \Rrightarrow \Omega_{\texttt{i}}$ while $\rho \vdash (\texttt{1} / \texttt{0}) < \texttt{X} \Rrightarrow \Omega_{\texttt{a}}$. However we will consider that all boolean expressions have been normalized (i.e. $B = T(B)$) because the respective evaluations of $B$ and $T(B)$ either produce the same boolean values (in general there is more than one possible value, because of random choice) or both expressions produce errors (which may be different). We have

$$\begin{gathered} \forall b \in \mathbb{B} : \rho \vdash B \Rrightarrow b \iff \rho \vdash T(B) \Rrightarrow b, \\ (\exists e \in \mathbb{E} : \rho \vdash B \Rrightarrow e) \iff (\exists e' \in \mathbb{E} : \rho \vdash T(B) \Rrightarrow e') \,. \end{gathered}$$

### 9.5 *Collecting semantics of boolean expressions*

The *collecting semantics* $\text{Cbexp}[\![B]\!]R$ of a boolean expression $B$ defines the subset of possible environments $\rho \in R$ for which the boolean expression may evaluate to true (hence without producing a runtime error)

$$\begin{aligned}\text{Cbexp} &\in \text{Bexp} \mapsto \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{R}),\\ \text{Cbexp}[\![B]\!]R &\triangleq \{\rho \in R \mid \rho \vdash B \Mapsto \text{tt}\}\ .\end{aligned} \tag{50}$$

## 10. Abstract Interpretation of Boolean Expressions

### 10.1 Generic abstract interpretation of boolean expressions

We now consider the calculational design of the generic nonrelational abstract semantics of boolean expressions

$$\text{Abexp} \in \text{Bexp} \mapsto (\mathbb{V} \mapsto L) \xrightarrow{\text{mon}} (\mathbb{V} \mapsto L)\ .$$

For any possible approximation (9) of value properties, this consists in approximating environment properties by the nonrelational abstraction (20) and in applying the following functional abstraction to the collecting semantics (50).

$$\langle \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{R}),\ \dot{\subseteq}\rangle \underset{\ddot{\alpha}}{\overset{\ddot{\gamma}}{\leftrightarrows}} \langle(\mathbb{V} \mapsto L) \xrightarrow{\text{mon}} (\mathbb{V} \mapsto L),\ \ddot{\sqsubseteq}\rangle \tag{51}$$

where

$$\begin{aligned}\Phi \mathrel{\dot{\subseteq}} \Psi &\triangleq \forall R \in \wp(\mathbb{R}) : \Phi(R) \subseteq \Psi(R),\\ \varphi \mathrel{\ddot{\sqsubseteq}} \psi &\triangleq \forall r \in \mathbb{V} \mapsto L : \varphi(r) \mathrel{\dot{\sqsubseteq}} \psi(r),\\ \ddot{\alpha}(\Phi) &\triangleq \dot{\alpha} \circ \Phi \circ \dot{\gamma},\\ \ddot{\gamma}(\varphi) &\triangleq \dot{\gamma} \circ \varphi \circ \dot{\alpha}\ .\end{aligned} \tag{52}$$

We must get an overapproximation such that

$$\text{Abexp}[\![B]\!] \mathrel{\ddot{\sqsupseteq}} \ddot{\alpha}(\text{Cbexp}[\![B]\!])\ . \tag{53}$$

We derive Abexp$[\![B]\!]$ as follows

$$\begin{aligned}&\ddot{\alpha}(\text{Cbexp}[\![B]\!])\\ =\ &\quad \wr\text{def. (52) of } \ddot{\alpha}\int\\ &\lambda r \in \mathbb{V} \mapsto L\bullet\dot{\alpha}(\text{Cbexp}[\![B]\!]\dot{\gamma}(r))\\ =\ &\quad \wr\text{def. (50) of Cbexp}\int\\ &\lambda r \in \mathbb{V} \mapsto L\bullet\dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash B \Mapsto \text{tt}\})\ .\end{aligned}$$

If $r$ is the infimum $\lambda\mathsf{Y}\bullet\bot$ and the infimum $\bot$ of $L$ is such that $\gamma(\bot) = \emptyset$ then $\dot{\gamma}(r) = \emptyset$. In this case

$$\begin{aligned}&\ddot{\alpha}(\text{Cbexp}[\![B]\!]\,\lambda\mathsf{Y}\bullet\bot)\\ =\ &\quad \wr\text{def. (19) of } \dot{\gamma}\int\\ &\dot{\alpha}(\emptyset)\\ =\ &\quad \wr\text{def. (18) of } \dot{\alpha}\int\\ &\lambda\mathsf{Y}\bullet\bot\ .\end{aligned}$$

Otherwise $r \neq \lambda\mathsf{Y}\bullet\bot$ or $\gamma(\bot) \neq \emptyset$, we have

$$\begin{aligned}&\ddot{\alpha}(\text{Cbexp}[\![B]\!])r\\ =\ &\quad \wr\text{def. lambda expression}\int\\ &\dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash B \Mapsto \text{tt}\}\ ,\end{aligned}$$

and we proceed by induction on the boolean expression $B$.

1 — When $B = \texttt{true}$ is true, we have

$$
\begin{array}{ll}
& \ddot{\alpha}(\mathrm{Cbexp}[\![\texttt{true}]\!])r \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash \texttt{true} \Rightarrow \mathrm{tt}\} \\
= & \quad \wr \text{def. (47) of } \rho \vdash \texttt{true} \Rightarrow b \wr \\
& \dot{\alpha}(\dot{\gamma}(r)) \\
\dot{\sqsubseteq} & \quad \wr \dot{\alpha} \circ \dot{\gamma} \text{ is reductive (51), (7)} \wr \\
& r \\
= & \quad \wr \text{by defining } \mathrm{Abexp}[\![\texttt{true}]\!]r \triangleq r \wr \\
& \mathrm{Abexp}[\![\texttt{true}]\!]r\ .
\end{array}
$$

2 — When $B = \texttt{false}$ is false, we have

$$
\begin{array}{ll}
& \ddot{\alpha}(\mathrm{Cbexp}[\![\texttt{false}]\!])r \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash \texttt{false} \Rightarrow \mathrm{tt}\} \\
= & \quad \wr \text{def. (48) of } \rho \vdash \texttt{false} \Rightarrow b \wr \\
& \dot{\alpha}(\emptyset) \\
= & \quad \wr \text{def. (18) of } \dot{\alpha} \wr \\
& \lambda \mathtt{Y} \boldsymbol{\cdot} \bot \\
= & \quad \wr \text{by defining } \mathrm{Abexp}[\![\texttt{false}]\!]r \triangleq \lambda \mathtt{Y} \boldsymbol{\cdot} \bot \wr \\
& \mathrm{Abexp}[\![\texttt{false}]\!]r\ .
\end{array}
$$

3 — When $B = A_1 \; \mathtt{c} \; A_2$ is an arithmetic comparison, we have

$$
\begin{array}{ll}
& \ddot{\alpha}(\mathrm{Cbexp}[\![A_1 \; \mathtt{c} \; A_2]\!])r \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \rho \vdash A_1 \; \mathtt{c} \; A_2 \Rightarrow \mathrm{tt}\}) \\
= & \quad \wr \text{def. (49) of } \rho \vdash A_1 \; \mathtt{c} \; A_2 \Rightarrow b \wr \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1, v_2 \in \mathbb{I}_\Omega : \rho \vdash A_1 \Rightarrow v_1 \wedge \rho \vdash A_2 \Rightarrow v_2 \wedge v_1 \; \underline{\mathrm{c}} \; v_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{set theory and } \gamma \circ \alpha \text{ is extensive (6)} \wr \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma(\alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_1 \Rightarrow v\})) : \\
& \qquad \exists v_2 \in \gamma(\alpha(\{v \mid \exists \rho \in \dot{\gamma}(r) : \rho \vdash A_2 \Rightarrow v\})) : \\
& \qquad\quad \rho \vdash A_1 \Rightarrow v_1 \wedge \rho \vdash A_2 \Rightarrow v_2 \wedge v_1 \; \underline{\mathrm{c}} \; v_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{set theory and (33)} \wr \\
= & \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r) : \exists v_2 \in \gamma(\mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) : \\
& \quad \rho \vdash A_1 \Rightarrow v_1 \wedge \rho \vdash A_2 \Rightarrow v_2 \wedge v_1 \; \underline{\mathrm{c}} \; v_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{let notation} \wr \\
& \text{let } \langle p_1,\ p_2 \rangle = \langle \mathrm{Faexp}^{\triangleright}[\![A_1]\!]r,\ \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r \rangle \text{ in} \\
& \quad \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists v_1 \in \gamma(p_1) : \exists v_2 \in \gamma(p_2) : \\
& \qquad \rho \vdash A_1 \Rightarrow v_1 \wedge \rho \vdash A_2 \Rightarrow v_2 \wedge v_1 \; \underline{\mathrm{c}} \; v_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{def. (45) of } \underline{\mathrm{c}} \text{ implying } v_1, v_2 \notin \mathbb{E} = \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\} \wr \\
& \text{let } \langle p_1,\ p_2 \rangle = \langle \mathrm{Faexp}^{\triangleright}[\![A_1]\!]r,\ \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r \rangle \text{ in} \\
& \quad \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists i_1 \in \gamma(p_1) \cap \mathbb{I} : \exists i_2 \in \gamma(p_2) \cap \mathbb{I} : \\
& \qquad\quad \rho \vdash A_1 \Rightarrow i_1 \wedge \rho \vdash A_2 \Rightarrow i_2 \wedge i_1 \; \underline{\mathrm{c}} \; i_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{set theory} \wr \\
& \text{let } \langle p_1,\ p_2 \rangle = \langle \mathrm{Faexp}^{\triangleright}[\![A_1]\!]r,\ \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r \rangle \text{ in} \\
& \quad \dot{\alpha}(\{\rho \in \dot{\gamma}(r) \mid \exists \langle i_1,\ i_2 \rangle \in \{\langle i_1',\ i_2' \rangle \mid i_1' \in \gamma(p_1) \cap \mathbb{I} \wedge i_2' \in \gamma(p_2) \cap \mathbb{I} \wedge i_1' \; \underline{\mathrm{c}} \; i_2' = \mathrm{tt}\} : \\
& \qquad\quad \rho \vdash A_1 \Rightarrow i_1 \wedge \rho \vdash A_2 \Rightarrow i_2 \wedge \}) \\
\dot{\sqsubseteq} & \quad \wr \gamma^2 \circ \alpha^2 \text{ extensive (13), (6) and } \dot{\alpha} \text{ monotone (20), (5)} \wr
\end{array}
$$

$$
\begin{array}{ll}
& \text{let } \langle p_1,\ p_2\rangle = \langle \mathrm{Faexp}^{\triangleright}[\![A_1]\!]r,\ \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r\rangle \text{ in} \\
& \quad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\alpha^2(\{\langle i'_1,\ i'_2\rangle \mid i'_1 \in \gamma(p_1)\cap\mathbb{I} \wedge i'_2 \in \gamma(p_2)\cap\mathbb{I} \wedge \\
& \qquad\qquad i'_1 \underline{\mathsf{c}}\, i'_2 = \mathrm{tt}\})) : \\
& \qquad\qquad \rho \vdash A_1 \Rrightarrow i_1 \wedge \rho \vdash A_2 \Rrightarrow i_2 \wedge\}) \\
\dot\sqsubseteq & \quad \wr \text{defining } \check{\mathsf{c}} \text{ such that:} \\
& \qquad \check{\mathsf{c}}(p_1, p_2) \sqsupseteq^2 \alpha^2(\{\langle i'_1,\ i'_2\rangle \mid i'_1 \in \gamma(p_1)\cap\mathbb{I} \wedge i'_2 \in \gamma(p_2)\cap\mathbb{I} \wedge i'_1 \underline{\mathsf{c}}\, i'_2 = \mathrm{tt}\}), \\
& \quad \gamma^2 \text{ and } \dot\alpha \text{ monotone (20), (5)} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \langle \mathrm{Faexp}^{\triangleright}[\![A_1]\!]r,\ \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r\rangle \text{ in} \\
& \quad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\check{\mathsf{c}}(p_1, p_2)) : \rho \vdash A_1 \Rrightarrow i_1 \wedge \rho \vdash A_2 \Rrightarrow i_2\}) \\
= & \quad \wr \text{let notation} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists\langle i_1,\ i_2\rangle \in \gamma^2(\langle p_1,\ p_2\rangle)) : \ \rho \vdash A_1 \Rrightarrow i_1 \wedge \rho \vdash A_2 \Rrightarrow i_2\}) \\
= & \quad \wr \text{set theory} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \gamma(p_1) : \rho \vdash A_1 \Rrightarrow i_1\} \cap \{\rho \in \dot\gamma(r) \mid \exists i_2 \in \gamma(p_2) : \rho \vdash A_2 \Rrightarrow i_2\}) \\
\dot\sqsubseteq & \quad \wr \dot\alpha \text{ monotone (20), (5)} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_1 \in \gamma(p_1) : \rho \vdash A_1 \Rrightarrow i_1\}) \,\dot\sqcap \\
& \qquad\qquad \dot\alpha(\{\rho \in \dot\gamma(r) \mid \exists i_2 \in \gamma(p_2) : \rho \vdash A_2 \Rrightarrow i_2\}) \\
= & \quad \wr \text{def. (29) of Baexp} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \dot\alpha(\mathrm{Baexp}[\![A_1]\!](\dot\gamma(r))\gamma(p_1)) \ \dot\sqcap\ \dot\alpha(\mathrm{Baexp}[\![A_2]\!](\dot\gamma(r))\gamma(p_2)) \\
= & \quad \wr \text{def. (37) of } \alpha^{\triangleleft} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \alpha^{\triangleleft}(\mathrm{Baexp}[\![A_1]\!])(r)p_1 \ \dot\sqcap\ \alpha^{\triangleleft}(\mathrm{Baexp}[\![A_2]\!])(r)p_2 \\
\dot\sqsubseteq & \quad \wr \text{def. (38) of } \mathrm{Baexp}^{\triangleleft} \text{ and } \dot\sqcap \text{ monotone} \wr \\
& \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \\
& \quad \mathrm{Baexp}^{\triangleleft}[\![A_1]\!](r)p_1 \ \dot\sqcap\ \mathrm{Baexp}^{\triangleleft}[\![A_2]\!](r)p_2 \\
= & \quad \wr \text{by defining } \mathrm{Abexp}[\![A_1 \ \mathsf{c}\ A_2]\!]r \triangleq \ \text{let } \langle p_1,\ p_2\rangle = \check{\mathsf{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in} \wr \\
& \qquad\qquad\qquad \mathrm{Baexp}^{\triangleleft}[\![A_1]\!](r)p_1 \ \dot\sqcap\ \mathrm{Baexp}^{\triangleleft}[\![A_2]\!](r)p_2 \\
& \mathrm{Abexp}[\![A_1 \ \mathsf{c}\ A_2]\!]r\ .
\end{array}
$$

4 — When $B = B_1 \,\&\, B_2$ is a conjunction, we have

$$
\begin{array}{ll}
& \ddot\alpha(\mathrm{Cbexp}[\![B_1 \,\&\, B_2]\!])r \\
= & \dot\alpha(\{\rho \in \dot\gamma(r) \mid \rho \vdash B_1 \Rrightarrow w_1 \wedge \rho \vdash B_2 \Rrightarrow w_2 \wedge w_1 \,\underline{\&}\, w_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{def. (46) of } \underline{\&} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \rho \vdash B_1 \Rrightarrow \mathrm{tt} \wedge \rho \vdash B_2 \Rrightarrow \mathrm{tt}\}) \\
= & \quad \wr \text{set theory} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \rho \vdash B_1 \Rrightarrow \mathrm{tt}\} \cap \{\rho \in \dot\gamma(r) \rho \vdash B_2 \Rrightarrow \mathrm{tt}\}) \\
\dot\sqsubseteq & \quad \wr \dot\alpha \text{ monotone (20), (5)} \wr \\
& \dot\alpha(\{\rho \in \dot\gamma(r) \mid \rho \vdash B_1 \Rrightarrow \mathrm{tt}\}) \ \dot\sqcap\ \dot\alpha(\{\rho \in \dot\gamma(r) \rho \vdash B_2 \Rrightarrow \mathrm{tt}\}) \\
= & \quad \wr \text{def. (50) of Cbexp} \wr \\
& \dot\alpha(\mathrm{Cbexp}[\![B_1]\!]\dot\gamma(r)) \ \dot\sqcap\ \dot\alpha(\mathrm{Cbexp}[\![B_2]\!]\dot\gamma(r)) \\
= & \quad \wr \text{def. (52) of } \ddot\alpha \wr \\
& \ddot\alpha(\mathrm{Cbexp}[\![B_1]\!])r \ \dot\sqcap\ \ddot\alpha(\mathrm{Cbexp}[\![B_2]\!])r \\
\dot\sqsubseteq & \quad \wr \text{induction hypothesis (53) and } \dot\sqcap \text{ monotone} \wr \\
& \mathrm{Abexp}[\![B_1]\!]r \ \dot\sqcap\ \mathrm{Abexp}[\![B_2]\!]r \\
= & \quad \wr \text{by defining } \mathrm{Abexp}[\![B_1 \,\&\, B_2]\!]r \triangleq \mathrm{Abexp}[\![B_1]\!]r \ \dot\sqcap\ \mathrm{Abexp}[\![B_2]\!]r \wr
\end{array}
$$

$$
\begin{aligned}
\mathrm{Abexp}[\![B]\!]\,\lambda\mathrm{Y}\bullet\bot &\stackrel{\Delta}{=} \lambda\mathrm{Y}\bullet\bot \quad \text{if} \quad \gamma(\bot)=\emptyset && (54)\\
\mathrm{Abexp}[\![\mathtt{true}]\!]r &\stackrel{\Delta}{=} r\\
\mathrm{Abexp}[\![\mathtt{false}]\!]r &\stackrel{\Delta}{=} \lambda\mathrm{Y}\bullet\bot\\
\mathrm{Abexp}[\![A_1 \mathtt{c} A_2]\!]r &\stackrel{\Delta}{=} \text{let } \langle p_1,\ p_2\rangle = \check{\mathtt{c}}(\mathrm{Faexp}^{\triangleright}[\![A_1]\!]r, \mathrm{Faexp}^{\triangleright}[\![A_2]\!]r) \text{ in}\\
&\qquad \mathrm{Baexp}^{\triangleleft}[\![A_1]\!](r)\,p_1 \ \dot{\sqcap}\ \mathrm{Baexp}^{\triangleleft}[\![A_2]\!](r)\,p_2\\
\mathrm{Abexp}[\![B_1 \,\&\, B_2]\!]r &\stackrel{\Delta}{=} \mathrm{Abexp}[\![B_1]\!]r \ \dot{\sqcap}\ \mathrm{Abexp}[\![B_2]\!]r\\
\mathrm{Abexp}[\![B_1 \mid B_2]\!]r &\stackrel{\Delta}{=} \mathrm{Abexp}[\![B_1]\!]r \ \dot{\sqcup}\ \mathrm{Abexp}[\![B_2]\!]r
\end{aligned}
$$

parameterized by the following abstract comparison operations $\check{\mathtt{c}}$, $\mathtt{c} \in \{<, =\}$ on $L$

$$
\check{\mathtt{c}}(p_1, p_2) \quad \sqsupseteq^2 \quad \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(p_1)\cap\mathbb{I} \wedge i_2 \in \gamma(p_2)\cap\mathbb{I} \wedge i_1\ \underline{\mathtt{c}}\ i_2 = \mathrm{tt}\})
$$

Figure 10: Abstract interpretation of boolean expressions

$\mathrm{Abexp}[\![B_1 \,\&\, B_2]\!]r$ .

5 —— The case $B = B_1 \mid B_2$ of disjunction is similar.

In conclusion, we have designed the abstract interpretation Abexp of boolean expressions in such a way that it satisfies the soundness requirement (53) as summarized in Fig. 10.

By induction on $B$, the operator $\mathrm{Abexp}[\![B]\!]$ on $\mathbb{V} \mapsto L$ is $\dot{\sqsubseteq}$-reductive and monotonic.

### *10.2 Generic static analyzer of boolean expressions*

The abstract comparison operations must be provided with the module implementing each particular algebra of abstract properties, as follows

```
module type Abstract_Lattice_Algebra_signature =
  sig
    (* complete lattice of abstract properties of values          *)
    type lat                                   (* abstract properties  *)
    ...
    (* forward abstract interpretation of arithmetic expressions  *)
    ...
    (* backward abstract interpretation of arithmetic expressions *)
    ...
    (* abstract interpretation of boolean expressions             *)
    val a_EQ : lat -> lat -> lat * lat
    val a_LT : lat -> lat -> lat * lat
  end;;
```

A functional implementation of Fig. 10 is

```
module Abexp_implementation =
 functor (L: Abstract_Lattice_Algebra_signature) ->
 functor (E: Abstract_Env_Algebra_signature) ->
 functor (Faexp: Faexp_signature) ->
 functor (Baexp: Baexp_signature) ->
 struct
  open Abstract_Syntax
```

```
(* generic abstract environments                                     *)
module E' = E (L)
(* generic forward abstract interpretation of arithmetic operations  *)
module Faexp' = Faexp(L)(E)
(* generic backward abstract interpretation of arithmetic operations *)
module Baexp' = Baexp(L)(E)(Faexp)
(* generic abstract interpretation of boolean operations             *)
let rec abexp' b r =
  match b with
  | TRUE           -> r
  | FALSE          -> (E'.bot ())
  | (EQ (a1, a2))  ->
      let (p1,p2) = (L.a_EQ (Faexp'.faexp a1 r) (Faexp'.faexp a2 r))
        in (E'.meet (Baexp'.baexp a1 r p1) (Baexp'.baexp a2 r p2))
  | (LT (a1, a2))  ->
      let (p1,p2) = (L.a_LT (Faexp'.faexp a1 r) (Faexp'.faexp a2 r))
        in (E'.meet (Baexp'.baexp a1 r p1) (Baexp'.baexp a2 r p2))
  | (AND (b1, b2)) -> (E'.meet (abexp' b1 r) (abexp' b2 r))
  | (OR (b1, b2))  -> (E'.join (abexp' b1 r) (abexp' b2 r))
let abexp b r =
  if (E'.is_bot r) & (L.isbotempty ()) then (E'.bot ()) else abexp' b r
end;;
```

### *10.3 Generic abstract boolean equality*

The calculational design of the abstract equality operation $\check{\doteq}$ does not depend upon the specific choice of $L$

$$
\begin{array}{ll}
 & \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(p_1) \cap \mathbb{I} \wedge i_2 \in \gamma(p_2) \cap \mathbb{I} \wedge i_1 \underline{=} i_2 = \mathrm{tt}\}) \\
 & \quad \wr\text{def. (45) of } \underline{=}\wr \\
 & \alpha^2(\{\langle i,\ i\rangle \mid i \in \gamma(p_1) \cap \gamma(p_2) \cap \mathbb{I}\}) \\
\sqsubseteq^2 & \quad \wr\gamma \circ \alpha \text{ is extensive (6) and } \alpha^2 \text{ is monotone}\wr \\
 & \alpha^2(\{\langle i,\ i\rangle \mid i \in \gamma(p_1) \cap \gamma(p_2) \cap \gamma(\alpha(\mathbb{I}))\}) \\
 & \quad \wr\gamma \text{ preserves meets}\wr \\
 & \alpha^2(\{\langle i,\ i\rangle \mid i \in \gamma(p_1 \sqcap p_2 \sqcap \alpha(\mathbb{I}))\}) \\
 & \quad \wr\text{def. (12) of } \gamma^2\wr \\
 & \alpha^2(\gamma^2(\langle p_1 \sqcap p_2 \sqcap \alpha(\mathbb{I}),\ p_1 \sqcap p_2 \sqcap \alpha(\mathbb{I})\rangle)) \\
\sqsubseteq^2 & \\
 & \quad \wr\alpha^2 \circ \gamma^2 \text{ is reductive and let notation}\wr \\
 & \text{let } p = p_1 \sqcap p_2 \sqcap \alpha(\mathbb{I}) \text{ in } \langle p,\ p\rangle \\
\sqsubseteq^2 & \quad \wr\text{def. (36) of } ?^{\triangleright}\wr \\
 & \text{let } p = p_1 \sqcap p_2 \sqcap ?^{\triangleright} \text{ in } \langle p,\ p\rangle \\
= & \quad \wr\text{by defining } \check{\doteq} \stackrel{\Delta}{=} \text{let } p = p_1 \sqcap p_2 \sqcap ?^{\triangleright} \text{ in } \langle p,\ p\rangle\wr \\
 & \check{\doteq}\ .
\end{array}
$$

In conclusion

$$p_1 \check{\doteq} p_2 \quad \stackrel{\Delta}{=} \quad \text{let } p = p_1 \sqcap p_2 \sqcap ?^{\triangleright} \text{ in } \langle p,\ p\rangle\ .$$

### *10.4 Initialization and simple sign abstract arithmetic comparison operations*

The abstract strict comparison

$$\check{<}(p_1, p_2) \quad \sqsupseteq^2 \quad \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(p_1) \cap \mathbb{I} \wedge i_2 \in \gamma(p_2) \cap \mathbb{I} \wedge i_1 \underline{<} i_2 = \mathrm{tt}\}) \qquad (55)$$

for initialization and simple sign analysis is as follows

| $\check{\lesssim}(p_1, p_2)$ | | $p_2$ | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| | | BOT, ERR | NEG | ZERO | POS | INI, TOP |
| $p_1$ | BOT, ERR | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ |
| | NEG | ⟨BOT, BOT⟩ | ⟨NEG, NEG⟩ | ⟨NEG, ZERO⟩ | ⟨NEG, POS⟩ | ⟨NEG, INI⟩ |
| | ZERO | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨ZERO, POS⟩ | ⟨ZERO, POS⟩ |
| | POS | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨BOT, BOT⟩ | ⟨POS, POS⟩ | ⟨POS, POS⟩ |
| | INI, TOP | ⟨BOT, BOT⟩ | ⟨NEG, NEG⟩ | ⟨NEG, ZERO⟩ | ⟨INI, POS⟩ | ⟨INI, INI⟩ |

Let us consider a few typical cases.

— If $p_i \in \{\texttt{BOT}, \texttt{ERR}\}$ where $i = 1$ or $i = 2$ then $\gamma(p_i) \subseteq \mathbb{E} = \{\Omega_{\mathtt{i}}, \Omega_{\mathtt{a}}\}$ so that $\gamma(p_i) \cap \mathbb{I} = \emptyset$ and we get:

$$\begin{array}{ll}
 & \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(p_1) \cap \mathbb{I} \wedge i_2 \in \gamma(p_2) \cap \mathbb{I} \wedge i_1 \underline{\leq} i_2 = \mathrm{tt}\}) \\
= & \alpha^2(\emptyset) \\
= & \quad \wr \text{def. (11) of } \alpha^2 \text{ and (15) of } \alpha \int \\
 & \langle \texttt{BOT},\ \texttt{BOT}\rangle \\
\triangleq & \quad \wr \text{def. (55) of } \check{\lesssim} \int \\
 & \check{\lesssim}(\texttt{BOT}, \texttt{BOT})\ ;
\end{array}$$

— For $\langle \texttt{POS},\ \texttt{ZERO}\rangle$, we have

$$\begin{array}{ll}
 & \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(\texttt{POS}) \cap \mathbb{I} \wedge i_2 \in \gamma(\texttt{ZERO}) \cap \mathbb{I} \wedge i_1 \underline{\leq} i_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{def. (14) of } \gamma \text{ and (45) of } \underline{\leq} \int \\
= & \alpha^2(\{\langle i_1,\ 0\rangle \mid i_1 \in [1, \texttt{max_int}] \wedge i_1 \leq 0\}) \\
= & \quad \wr \text{set theory} \int \\
 & \alpha^2(\emptyset) \\
= & \quad \wr \text{def. (11) of } \alpha^2 \text{ and (15) of } \alpha \int \\
 & \langle \texttt{BOT},\ \texttt{BOT}\rangle \\
\triangleq & \quad \wr \text{def. (55) of } \check{\lesssim} \int \\
 & \check{\lesssim}(\texttt{POS}, \texttt{ZERO})\ .
\end{array}$$

— For $\langle \texttt{TOP},\ \texttt{TOP}\rangle$, we have

$$\begin{array}{ll}
 & \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \gamma(\texttt{TOP}) \cap \mathbb{I} \wedge i_2 \in \gamma(\texttt{TOP}) \cap \mathbb{I} \wedge i_1 \underline{\leq} i_2 = \mathrm{tt}\}) \\
= & \quad \wr \text{def. (14) of } \gamma \text{ and (45) of } \underline{\leq} \int \\
\mathrm{s} & \alpha^2(\{\langle i_1,\ i_2\rangle \mid i_1 \in \mathbb{I} \wedge i_2 \in \mathbb{I} \wedge i_1 \leq i_2\}) \\
= & \quad \wr \text{def. (11) of } \alpha^2 \int \\
 & \langle \alpha(\mathbb{I}),\ \alpha(\mathbb{I})\rangle \\
= & \quad \wr \text{def. (15) of } \alpha \int \\
 & \langle \texttt{INI},\ \texttt{INI}\rangle \\
\triangleq & \quad \wr \text{def. (55) of } \check{\lesssim} \int \\
 & \check{\lesssim}(\texttt{TOP}, \texttt{TOP})\ .
\end{array}$$

## 11. Reductive Iteration

### *11.1 Iterating monotone and reductive abstract operators*

The idea of iterating a monotone and reductive abstract operator to get a more precise lower closure abstract operator was used to define the "reduced product" of [13] and in the "local decreasing iterations" examples of [24] to handle backward assignments and conditionals (the same idea was later exploited in logic program analysis under the name of "reexecution" [29]). More generally, the idea is that of reductive iterations.

**Theorem 1** *If $\langle M, \preceq\rangle$ is poset, $f \in M \mapsto M$ is monotone and reductive, $\langle M, \preceq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L, \sqsubseteq\rangle$ is a Galois connection, $\langle L, \sqsubseteq, \sqcap\rangle$ is a dual dcpo, $g \in L \mapsto L$ is monotone and reductive and $\alpha \circ f \circ \gamma \mathrel{\dot\sqsubseteq} g$ then the lower closure operator $g^\star \triangleq \lambda x \cdot gfp_x^{\sqsubseteq}\, g$ is a better abstract interpretation of $f$ than $g$*

$$\alpha \circ f \circ \gamma \quad \dot\sqsubseteq \quad g^\star \quad \dot\sqsubseteq \quad g \ .$$

*Proof* For all $x \in L$, $g(x) \sqsubseteq x$, so that by monotony the sequence $g^0(x) \triangleq x$, $g^{\delta+1}(x) \triangleq g(g^\delta(x))$ for all successor ordinals $\delta + 1$ and $g^\lambda \triangleq \bigsqcap_{\delta<\lambda} g^\delta(x)$ for all limit ordinals $\lambda$ is a well-defined decreasing chain in the dual dcpo $\langle L, \sqsubseteq, \sqcap\rangle$ whence ultimately stationary. It converges to $g^\epsilon$ where $\epsilon$ is the order of $g$, which is the greatest fixpoint $g^\epsilon = \mathrm{gfp}_x^{\sqsubseteq}\, g$ of $g$ which is $\sqsubseteq$-less than $x$ [12]. It follows that $g^\star \triangleq \mathrm{gfp}_x^{\sqsubseteq}\, g$ is the greatest lower closure operator $\dot\sqsubseteq$-less than $g$ [11]. In particular $g^\star \mathrel{\dot\sqsubseteq} g$.

We have $\alpha \circ f \circ \gamma(x) \sqsubseteq g^1(x) = g(x) \sqsubseteq x = g^0(x)$. If $\alpha \circ f \circ \gamma(x) \sqsubseteq g^\delta(x)$ then

$$
\begin{array}{ll}
 & \alpha \circ f \circ \gamma(x) \\
\sqsubseteq & \quad \wr f \text{ reductive (so that } f(f(\gamma(x))) \sqsubseteq f(\gamma(x))\text{) and } \alpha \text{ monotone} \int \\
 & \alpha \circ f \circ f \circ \gamma(x) \\
\sqsubseteq & \quad \wr \gamma \circ \alpha \text{ is extensive, } f \text{ and } \alpha \text{ are monotone} \int \\
 & \alpha \circ f \circ \gamma \circ \alpha \circ f \circ \gamma(x) \\
\sqsubseteq & \quad \wr \alpha \circ f \circ \gamma(x) \sqsubseteq g^\delta(x) \text{ by induction hypothesis, } \gamma,\ f \text{ and } \alpha \text{ are monotone} \int \\
 & \alpha \circ f \circ \gamma(g^\delta(x)) \\
\sqsubseteq & \quad \wr \alpha \circ f \circ \gamma \mathrel{\dot\sqsubseteq} g \text{ hypothesis} \int \\
 & g(g^\delta(x)) \\
= & \quad \wr \text{def. } g^{\delta+1}(x) \int \\
 & g^{\delta+1}(x) \ .
\end{array}
$$

If $\alpha \circ f \circ \gamma(x) \sqsubseteq g^\delta(x)$ for all $\delta < \lambda$ and $\lambda$ is a limit ordinal then by definition of lubs and $g^\lambda$, we have $\alpha \circ f \circ \gamma(x) \sqsubseteq \bigsqcap_{\delta<\lambda} g^\delta(x) = g^\lambda$. By transfinite induction, $\alpha \circ f \circ \gamma(x) \sqsubseteq g^\epsilon(x) = g^\star(x)$. □

*11.2 Reductive iteration for boolean and arithmetic expressions*

Reductive iteration has a direct application to the analysis of boolean expressions. The abstract interpretation $\mathrm{Abexp}[\![B]\!]$ of boolean expressions $B$ defined in Sec. 10.1 can always be replaced by its reductive iteration $\mathrm{Abexp}[\![B]\!]^\star$ which is sound (53) and always more precise. By Th. 1, we have

$$\ddot\alpha(\mathrm{Cbexp}[\![B]\!]) \quad \ddot\sqsubseteq \quad \mathrm{Abexp}[\![B]\!]^\star \quad \ddot\sqsubseteq \quad \mathrm{Abexp}[\![B]\!] \ .$$

The same way, for the backward analysis of arithmetic expressions of Sec. 8.5, we have:

$$\forall p \in L : \lambda r \cdot \alpha^\triangleleft(\mathrm{Baexp}[\![A]\!])(r)p \quad \ddot\sqsubseteq \quad (\lambda r \cdot \mathrm{Baexp}^\triangleleft[\![A]\!](r)p)^\star \quad \ddot\sqsubseteq \quad \lambda r \cdot \mathrm{Baexp}^\triangleleft[\![A]\!](r)p \ .$$

### 11.3 Generic implementation of reductive iteration

The implementation of reductive iteration is based upon a fixpoint computation over posets (satisfying the ascending and descending chain conditions), as follows

```
module type Poset_signature =
  sig
    type element
    val leq : element -> element -> bool
  end;;

module type Fixpoint_signature =
  functor (P:Poset_signature) ->
  sig
    val lfp : (P.element -> P.element) -> P.element -> P.element
    val gfp : (P.element -> P.element) -> P.element -> P.element
  end;;

module Fixpoint_implementation =
  functor (P:Poset_signature) ->
  struct
    (* iterative computation of the least fixpoint of f greater *)
    (* than or equal to the prefixpoint x (f(x) >= x)           *)
    let rec lfp f x =
      let x' = (f x) in
        if (P.leq x' x) then x'
        else lfp f x'
    (* iterative computation of the greatest fixpoint of f less *)
    (* than or equal to the postfixpoint x (f(x) <= x)          *)
    let rec gfp f x =
      let x' = (f x) in
        if (P.leq x x') then x
else gfp f x'
  end;;

module Fixpoint = (Fixpoint_implementation:Fixpoint_signature);;
```

For abstract domains $L$ not satisfying the descending chain condition, a narrowing operator [9] must be used to ensure convergence to an overapproximation. The implementation of reductive iteration is then straightforward.

```
module Baexp_Reductive_Iteration_implementation =
  functor (Baexp: Baexp_signature) ->
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  functor (Faexp: Faexp_signature) ->
  struct
    (* generic abstract elementironments *)
    module E' = E (L)
    (* iterative fixpoint computation *)
    module F = Fixpoint((E':Poset_signature with type element = E (L).env))
    (* generic backward abstract interpretation of arithmetic operations *)
    module Baexp' = Baexp(L)(E)(Faexp)
    (* generic reductive backward abstract int. of arithmetic operations *)
    let baexp a r p =
      let f x = Baexp'.baexp a x p in
        F.gfp f r
  end;;
```

```
module Baexp_Reductive_Iteration =
  (Baexp_Reductive_Iteration_implementation (Baexp):Baexp_signature);;
```

Either of the `Baexp` or `Baexp_Reductive_Iteration` modules can be used by (i.e. passed as parameters to) the generic static analyzer. Here is an example of reachability analysis where all variables are assumed to be uninitialized at the program entry point with the initialization and simple sign abstraction. Abstract invariants automatically derived by the analysis are written below in italic between round brackets.

without reductive iteration:

```
{ x:ERR; y:ERR; z:ERR }
  x := 0; y := ?; z := ?;
{ x:ZERO; y:INI; z:INI }
  if ((x=y)&(y=z)&((z+1)=x)) then
    { x:ZERO; y:ZERO; z:NEG }
      skip
  else
    { x:ZERO; y:INI; z:INI }
      skip
  fi
{ x:ZERO; y:INI; z:INI }
```

with reductive iteration:

```
{ x:ERR; y:ERR; z:ERR }
  x := 0; y := ?; z := ?;
{ x:ZERO; y:INI; z:INI }
  if ((x=y)&(y=z)&((z+1)=x)) then
    { x:BOT; y:BOT; z:BOT }
      skip
  else
    { x:ZERO; y:INI; z:INI }
      skip
  fi
{ x:ZERO; y:INI; z:INI }
```

Informally, without reductive iteration, from `{x:ZERO; y:INI}` and `(x=y)` we get `{x:ZERO; y:ZERO}`. Besides, from `{y:INI; z:INI}` and `(y=z)` we gain no information. Finally from `{x:ZERO; z:INI}` and `((z+1)=x)`, we get `{x:ZERO; z:NEG}`. By conjunction, we conclude with the invariant `{x:ZERO; y:ZERO; z:NEG}`. With reductive iteration, the analysis is repeated. So from `{y:ZERO; z:NEG}` and `(y=z)`, we reduce to `BOT`.

## 12. Semantics of Imperative Programs

### *12.1 Abstract syntax of commands and programs*

The abstract syntax of programs is given in Fig. 11.

### *12.2 Program components*

A program may be represented in abstract syntax as a finite ordered labelled tree, the leaves of which are labelled with identity commands, assignment commands and boolean expressions (which can themselves be represented by finite abstract syntax trees) and the internal nodes of which are labelled with conditional, iteration and sequence labels. Each subtree $\lfloor C \rfloor_\pi$, which uniquely identifies a component (subcommand or subsequence) $C$ of a program, can be designated by a *position* $\pi$, that is a sequence of positive integers — in Dewey decimal notation — , describing the path within the program abstract syntax tree from the outermost program root symbol to the head of the component at that position (which is standard in rewrite

*Variables*

$\mathtt{X} \in \mathbb{V}$ .

*Arithmetic expressions*

$A \in \text{Aexp}$ .

*Boolean expressions*

$B \in \text{Bexp}$ .

*Commands*

$$
\begin{array}{rcll}
C \in \text{Com} & ::= & \mathtt{skip} & \text{identity,} \\
& | & \mathtt{X} \;\mathtt{:=}\; A & \text{assignment,} \\
& | & \mathtt{if}\; B \;\mathtt{then}\; S_1 \;\mathtt{else}\; S_2 \;\mathtt{fi} & \text{conditional,} \\
& | & \mathtt{while}\; B \;\mathtt{do}\; S \;\mathtt{od} & \text{iteration.}
\end{array}
$$

*List of commands*

$$
\begin{array}{rcll}
S, S_1, S_2 \in \text{Seq} & ::= & C & \text{command,} \\
& | & C \;\mathtt{;}\; S & \text{sequence.}
\end{array}
$$

*Program*

$$
\begin{array}{rcll}
P \in \text{Prog} & ::= & S \;\mathtt{;;} & \text{program.}
\end{array}
$$

Figure 11: Abstract syntax of commands and programs

systems [21]). These program components are defined as follows:

$$
\begin{aligned}
\text{Cmp}[\![S \;\mathtt{;;}]\!] &\triangleq \{\lfloor S \;\mathtt{;;} \rfloor_0\} \cup \text{Cmp}^{O}[\![S]\!], \\
\text{Cmp}^{\pi}[\![C_1 \;\mathtt{;}\; \ldots \;\mathtt{;}\; C_n]\!] &\triangleq \{\lfloor C_1 \;\mathtt{;}\; \ldots \;\mathtt{;}\; C_n \rfloor_\pi\} \cup \bigcup_{i=1}^{n} \text{Cmp}^{\pi.i}[\![C_i]\!], \\
\text{Cmp}^{\pi}[\![\mathtt{if}\; B \;\mathtt{then}\; S_1 \;\mathtt{else}\; S_2 \;\mathtt{fi}]\!] &\triangleq \{\lfloor \mathtt{if}\; B \;\mathtt{then}\; S_1 \;\mathtt{else}\; S_2 \;\mathtt{fi} \rfloor_\pi\} \cup \text{Cmp}^{\pi.1}[\![S_1]\!] \cup \\
&\qquad \text{Cmp}^{\pi.2}[\![S_2]\!], \\
\text{Cmp}^{\pi}[\![\mathtt{while}\; B \;\mathtt{do}\; S_1 \;\mathtt{od}]\!] &\triangleq \{\lfloor \mathtt{while}\; B \;\mathtt{do}\; S_1 \;\mathtt{od} \rfloor_\pi\} \cup \text{Cmp}^{\pi.1}[\![S_1]\!], \\
\text{Cmp}^{\pi}[\![\mathtt{X} \;\mathtt{:=}\; A]\!] &\triangleq \{\lfloor \mathtt{X} \;\mathtt{:=}\; A \rfloor_\pi\}, \\
\text{Cmp}^{\pi}[\![\mathtt{skip}]\!] &\triangleq \{\lfloor \mathtt{skip} \rfloor_\pi\} .
\end{aligned}
$$

For example $\text{Cmp}[\![\mathtt{skip}\;\mathtt{;}\;\mathtt{skip}\;\mathtt{;;}]\!] = \{\lfloor \mathtt{skip}\;\mathtt{;}\;\mathtt{skip}\;\mathtt{;;} \rfloor_0, \lfloor \mathtt{skip} \rfloor_{01}, \lfloor \mathtt{skip} \rfloor_{02}\}$ so that the two occurrences of the same command **skip** within the program **skip** ; **skip** ;; can be formally distinguished.

## 12.3 Program labelling

In practice the above positions are not quite easy to use for identifying program components. We prefer labels $\ell \in \text{Lab}$ designating program points ($P \in \text{Prog}$)

$$
\begin{aligned}
\text{at}_P, \text{after}_P &\in \text{Cmp}[\![P]\!] \mapsto \text{Lab}, \\
\text{in}_P &\in \text{Cmp}[\![P]\!] \mapsto \wp(\text{Lab}) .
\end{aligned}
$$

Program components labelling is defined as follows (for short we leave positions implicit, writing $C$ for $\lfloor C \rfloor_\pi$ and assuming that the rules for designating subcomponents of a component are clear from Sec. 12.2)

$$\forall C \in \mathrm{Cmp}[\![P]\!] : \mathrm{at}_P[\![C]\!] \neq \mathrm{after}_P[\![C]\!] \ . \tag{56}$$

If $C = \texttt{skip} \in \mathrm{Cmp}[\![P]\!]$ or $C = \mathsf{X}\ \texttt{:=}\ A \in \mathrm{Cmp}[\![P]\!]$ then

$$\mathrm{in}_P[\![C]\!] \quad = \quad \{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \ . \tag{57}$$

If $S = C_1; \ldots ; C_n \in \mathrm{Cmp}[\![P]\!]$ where $n \geq 1$ is a sequence of commands, then

$$\begin{array}{rcl}
\mathrm{at}_P[\![S]\!] & = & \mathrm{at}_P[\![C_1]\!], \\
\mathrm{after}_P[\![S]\!] & = & \mathrm{after}_P[\![C_n]\!], \\
\mathrm{in}_P[\![S]\!] & = & \displaystyle\bigcup_{i=1}^{n} \mathrm{in}_P[\![C_i]\!], \\
\multicolumn{3}{l}{\forall i \in [1, n[: \mathrm{after}_P[\![C_i]\!] = \mathrm{at}_P[\![C_{i+1}]\!] = \mathrm{in}_P[\![C_i]\!] \cap \mathrm{in}_P[\![C_{i+1}]\!],} \\
\multicolumn{3}{l}{\forall i, j \in [1, n] : (j \neq i-1 \wedge j \neq i+1) \implies (\mathrm{in}_P[\![C_i]\!] \cap \mathrm{in}_P[\![C_j]\!] = \emptyset) \ .}
\end{array} \tag{58}$$

If $C = \texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi} \in \mathrm{Cmp}[\![P]\!]$ is a conditional command, then

$$\begin{array}{rcl}
\mathrm{in}_P[\![C]\!] & = & \{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cup \mathrm{in}_P[\![S_t]\!] \cup \mathrm{in}_P[\![S_f]\!], \\
\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cap (\mathrm{in}_P[\![S_t]\!] \cup \mathrm{in}_P[\![S_f]\!]) & = & \emptyset, \\
\mathrm{in}_P[\![S_t]\!] \cap \mathrm{in}_P[\![S_f]\!] & = & \emptyset \ .
\end{array} \tag{59}$$

If $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od} \in \mathrm{Cmp}[\![P]\!]$ is an iteration command, then

$$\begin{array}{l}
\mathrm{in}_P[\![C]\!] = \{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cup \mathrm{in}_P[\![S]\!], \\
\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cap \mathrm{in}_P[\![S]\!] = \emptyset \ .
\end{array} \tag{60}$$

If $P = S\ \texttt{;;} \in \mathrm{Cmp}[\![P]\!]$ is a program, then

$$\mathrm{at}_P[\![P]\!] \ = \ \mathrm{at}_P[\![S]\!], \qquad \mathrm{after}_P[\![P]\!] \ = \ \mathrm{after}_P[\![S]\!], \qquad \mathrm{in}_P[\![P]\!] \ = \ \mathrm{in}_P[\![S]\!] \ .$$

### *12.4 Program variables*

The free variables

$$\mathrm{Var} \quad \in \quad (\mathrm{Prog} \cup \mathrm{Com} \cup \mathrm{Seq} \cup \mathrm{Aexp} \cup \mathrm{Bexp}) \mapsto \wp(\mathbb{V})$$

are defined as usual for *programs* ($S \in \mathrm{Seq}$)

$$\mathrm{Var}[\![S\ \texttt{;;}]\!] \ \stackrel{\Delta}{=} \ \mathrm{Var}[\![S]\!] \ ;$$

*list of commands* ($C \in \mathrm{Com}$, $S \in \mathrm{Seq}$)

$$\mathrm{Var}[\![C\ \texttt{;}\ S]\!] \ \stackrel{\Delta}{=} \ \mathrm{Var}[\![C]\!] \cup \mathrm{Var}[\![S]\!] \ ;$$

*commands* ($\mathsf{X} \in \mathbb{V}$, $A \in \mathrm{Aexp}$, $B \in \mathrm{Bexp}$, $S, S_t, S_f \in \mathrm{Seq}$)

$$\begin{array}{rcl}
\mathrm{Var}[\![\texttt{skip}]\!] & \stackrel{\Delta}{=} & \emptyset, \\
\mathrm{Var}[\![\mathsf{X}\ \texttt{:=}\ A]\!] & \stackrel{\Delta}{=} & \{\mathsf{X}\} \cup \mathrm{Var}[\![A]\!], \\
\mathrm{Var}[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!] & \stackrel{\Delta}{=} & \mathrm{Var}[\![B]\!] \cup \mathrm{Var}[\![S_t]\!] \cup \mathrm{Var}[\![S_f]\!], \\
\mathrm{Var}[\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!] & \stackrel{\Delta}{=} & \mathrm{Var}[\![B]\!] \cup \mathrm{Var}[\![S]\!] \ ;
\end{array}$$

*arithmetic expressions* ($\mathtt{n} \in \mathrm{Nat}$, $\mathtt{X} \in \mathbb{V}$, $\mathtt{u} \in \{+, -\}$, $A_1, A_2 \in \mathrm{Aexp}$, $\mathtt{b} \in \{+, -, *, /, \mathtt{mod}\}$)

$$\begin{array}{rclrcl}
\mathrm{Var}[\![\mathtt{n}]\!] & \triangleq & \emptyset, & \mathrm{Var}[\![\mathtt{u}\ A_1]\!] & \triangleq & \mathrm{Var}[\![A_1]\!], \\
\mathrm{Var}[\![\mathtt{X}]\!] & \triangleq & \{\mathtt{X}\}, & \mathrm{Var}[\![A_1\ \mathtt{b}\ A_2]\!] & \triangleq & \mathrm{Var}[\![A_1]\!] \cup \mathrm{Var}[\![A_2]\!], \\
\mathrm{Var}[\![?]\!] & \triangleq & \emptyset & & &
\end{array}$$

and *boolean expressions* ($A_1, A_2 \in \mathrm{Aexp}$, $\mathtt{r} \in \{=, <\}$, $B_1, B_2 \in \mathrm{Bexp}$, $\mathtt{l} \in \{\&, |\}$)

$$\begin{array}{rclrcl}
\mathrm{Var}[\![\mathtt{true}]\!] & \triangleq & \emptyset, & \mathrm{Var}[\![A_1\ \mathtt{r}\ A_2]\!] & \triangleq & \mathrm{Var}[\![A_1]\!] \cup \mathrm{Var}[\![A_2]\!], \\
\mathrm{Var}[\![\mathtt{false}]\!] & \triangleq & \emptyset, & \mathrm{Var}[\![B_1\ \mathtt{l}\ B_2]\!] & \triangleq & \mathrm{Var}[\![B_1]\!] \cup \mathrm{Var}[\![B_2]\!] .
\end{array}$$

### *12.5 Program states*

During execution of program $P \in \mathrm{Prog}$, an environment $\rho \in \mathrm{Env}[\![P]\!] \subseteq \mathbb{R}$ maps program variables $\mathtt{X} \in \mathrm{Var}[\![P]\!]$ to their value $\rho(\mathtt{X})$. We define

$$\begin{array}{rcl}
\mathrm{Env} & \in & \mathrm{Prog} \mapsto \wp(\mathbb{R}), \\
\mathrm{Env}[\![P]\!] & \triangleq & \mathrm{Var}[\![P]\!] \mapsto \mathbb{I}_\Omega .
\end{array}$$

States $\langle \ell,\ \rho \rangle \in \Sigma[\![P]\!]$ record a program point $\ell \in \mathrm{in}_P[\![P]\!]$ and an environment $\rho \in \mathrm{Env}[\![P]\!]$ assigning values to variables. The definition of states is

$$\begin{array}{rcl}
\Sigma & \in & \mathrm{Prog} \mapsto \wp(\mathbb{V} \times \mathbb{R}), \\
\Sigma[\![P]\!] & \triangleq & \mathrm{in}_P[\![P]\!] \times \mathrm{Env}[\![P]\!] .
\end{array} \tag{61}$$

### *12.6 Small-step operational semantics of commands*

The small-step operational semantics [31] of commands, sequences and programs $C \in \mathrm{Com} \cup \mathrm{Seq} \cup \mathrm{Prog}$ within a program $P \in \mathrm{Prog}$ involves transition judgements

$$\langle \ell,\ \rho \rangle \models\!\!=[\![C]\!]\!\!\Longrightarrow \langle \ell',\ \rho' \rangle .$$

Such judgements mean that if execution is at control point $\ell \in \mathrm{in}_P[\![C]\!]$ in environment $\rho \in \mathrm{Env}[\![P]\!]$ then the next computation step within command $C$ leads to program control point $\ell' \in \mathrm{in}_P[\![C]\!]$ in the new environment $\rho' \in \mathrm{Env}[\![P]\!]$. The definition of $\langle \ell,\ \rho \rangle \models\!\!=[\![C]\!]\!\!\Longrightarrow \langle \ell',\ \rho' \rangle$ is by structural induction on $C$ as shown in Fig. 12.

According to axiom schema (63), program execution is blocked in error state at the assignment $\mathtt{X} := A$ if the arithmetic expression $A$ evaluates to an error, i.e. $\rho \vdash A \Mapsto \Omega_e$, $e \in \mathbb{E}$ [5]. The same way in conditional and iteration commands, execution is blocked when a boolean expression is erroneous i.e. evaluates to $\rho \vdash B \Mapsto \Omega_e$, $e \in \mathbb{E}$ [6]. Note that in the definition (74) of the small-step operational semantics of sequences, the proper sequencing directly follows from the labelling scheme (58) since $\mathrm{after}_P[\![C_i]\!] = \mathrm{at}_P[\![C_{i+1}]\!]$.

[5] This option corresponds to an implementation where uninitialization is implemented using a special value which is checked at runtime whenever a variable is used in arithmetic (or boolean) expressions.

[6] Another possible semantics would be a nondeterministic choice of the chosen branch. This option corresponds to an implementation where the initial variable can be any value.

*Identity* $C = \texttt{skip}$ ($\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$)

$$\langle \ell,\ \rho\rangle \Longmapsto[\![\texttt{skip}]\!]\Longrightarrow \langle \ell',\ \rho\rangle\ . \tag{62}$$

*Assignment* $C = \mathtt{X} := A$ ($\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$)

$$\frac{\rho \vdash A \Rrightarrow i}{\langle \ell,\ \rho\rangle \Longmapsto[\![\mathtt{X} := A]\!]\Longrightarrow \langle \ell',\ \rho[\mathtt{X} \leftarrow i]\rangle},\ i \in \mathbb{I}\ . \tag{63}$$

*Conditional* $C = \texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}$ ($\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$)

$$\frac{\rho \vdash B \Rrightarrow \mathrm{tt}}{\langle \ell,\ \rho\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle}\ , \tag{64}$$

$$\frac{\rho \vdash T(\neg B) \Rrightarrow \mathrm{tt}}{\langle \ell,\ \rho\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \mathrm{at}_P[\![S_f]\!],\ \rho\rangle}\ . \tag{65}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![S_t]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ , \tag{66}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![S_f]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ . \tag{67}$$

$$\langle \mathrm{after}_P[\![S_t]\!],\ \rho\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \ell',\ \rho\rangle\ , \tag{68}$$

$$\langle \mathrm{after}_P[\![S_f]\!],\ \rho\rangle \Longmapsto[\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\Longrightarrow \langle \ell',\ \rho\rangle\ . \tag{69}$$

*Iteration* $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}$ ($\mathrm{at}_P[\![C]\!] = \ell$, $\mathrm{after}_P[\![C]\!] = \ell'$ and $\ell_1, \ell_2 \in \mathrm{in}_P[\![S]\!]$)

$$\frac{\rho \vdash T(\neg B) \Rrightarrow \mathrm{tt}}{\langle \ell,\ \rho\rangle \Longmapsto[\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!]\Longrightarrow \langle \ell',\ \rho\rangle}\ , \tag{70}$$

$$\frac{\rho \vdash B \Rrightarrow \mathrm{tt}}{\langle \ell,\ \rho\rangle \Longmapsto[\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!]\Longrightarrow \langle \mathrm{at}_P[\![S]\!],\ \rho\rangle}\ , \tag{71}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![S]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \Longmapsto[\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!]\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ , \tag{72}$$

$$\langle \mathrm{after}_P[\![S]\!],\ \rho\rangle \Longmapsto[\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!]\Longrightarrow \langle \ell,\ \rho\rangle\ . \tag{73}$$

*Sequence* $C_1\ \texttt{;}\ \ldots\ \texttt{;}\ C_n, n > 0$ ($\ell_i, \ell_{i+1} \in \mathrm{in}_P[\![C_i]\!]$ for all $i \in [1, n]$)

$$\frac{\langle \ell_i,\ \rho_i\rangle \Longmapsto[\![C_i]\!]\Longrightarrow \langle \ell_{i+1},\ \rho_{i+1}\rangle}{\langle \ell_i,\ \rho_i\rangle \Longmapsto[\![C_1\ \texttt{;}\ \ldots\ \texttt{;}\ C_n]\!]\Longrightarrow \langle \ell_{i+1},\ \rho_{i+1}\rangle}\ . \tag{74}$$

*Program* $P = S\ \texttt{;;}$

$$\frac{\langle \ell,\ \rho\rangle \Longmapsto[\![S]\!]\Longrightarrow \rho}{\langle \ell',\ \rho'\rangle \Longmapsto[\![S\ \texttt{;;}]\!]\Longrightarrow \langle \ell',\ \rho'\rangle}\ . \tag{75}$$

Figure 12: Small-step operational semantics of commands and programs

## 12.7 *Transition system of a program*

The transition system of a program $P = S$ `;;` is

$$\langle \Sigma[\![P]\!],\ \tau[\![P]\!]\rangle$$

where $\Sigma[\![P]\!]$ is the set (61) of program states and $\tau[\![C]\!]$, $C \in \mathrm{Cmp}[\![P]\!]$ is the transition relation for component $C$ of program $P$, defined by

$$\tau[\![C]\!] \ \triangleq\ \{\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \mid \langle \ell,\ \rho\rangle \Longmapsto\!\![\![C]\!]\!\!\Longrightarrow \langle \ell',\ \rho'\rangle\}\ . \tag{76}$$

Execution starts at the program entry point with all variables uninitialized

$$\mathrm{Entry}[\![P]\!] \ \triangleq\ \{\langle \mathrm{at}_P[\![P]\!],\ \lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \boldsymbol{\cdot} \Omega_{\mathtt{i}}\rangle\}\ . \tag{77}$$

Execution ends without error when control reaches the program exit point

$$\mathrm{Exit}[\![P]\!] \ \triangleq\ \{\mathrm{after}_P[\![P]\!]\} \times \mathrm{Env}[\![P]\!]\ .$$

When the evaluation of an arithmetic or boolean expression fails with a runtime error, the program execution is blocked so that no further transition is possible.

A basic result on the program transition relation is that it is not possible to jump into or out of program components ($C \in \mathrm{Cmp}[\![P]\!]$)

$$\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau[\![C]\!] \quad \Longrightarrow \quad \{\ell, \ell'\} \subseteq \mathrm{in}_P[\![C]\!]\ . \tag{78}$$

The proof, by structural induction on $C$, is trivial whence omitted.

## 12.8 *Reflexive transitive closure of the program transition relation*

The reflexive transitive closure of the transition relation $\tau[\![C]\!]$ of a program component $C \in \mathrm{Cmp}[\![P]\!]$ is $\tau^\star[\![C]\!] \triangleq (\tau[\![C]\!])^\star$. $\tau^\star[\![P]\!]$ can be expressed compositionally (by structural induction the the components $C \in \mathrm{Cmp}[\![P]\!]$ of program $P$). The computational design follows.

1 — For the identity $C = \texttt{skip}$ and the assignment $C = \mathtt{X} := A$

$$\tau^\star[\![C]\!]$$

$=$ $\wr$def. of $(\tau[\![C]\!])^\star$ and $\tau[\![C]\!]$ so that $\mathrm{at}_P[\![C]\!] \neq \mathrm{after}_P[\![C]\!]$ by (56) implies $(\tau[\![C]\!])^2 = \emptyset$, whence by recurrence $(\tau[\![C]\!])^n = \emptyset$ for all $n \geq 2$, $1_S$ was defined as the identity on the set $S\wr$

$$1_{\Sigma[\![P]\!]} \cup \tau[\![C]\!]\ .$$

2 — For the conditional $C = \texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}$, we define

$$\begin{aligned}
\tau^B &\ \triangleq\ \{\langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Rightarrow \mathrm{tt}\},\\
\tau^{\bar{B}} &\ \triangleq\ \{\langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_f]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Rightarrow \mathrm{tt}\},\\
\tau^t &\ \triangleq\ \{\langle\langle \mathrm{after}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\},\\
\tau^f &\ \triangleq\ \{\langle\langle \mathrm{after}_P[\![S_f]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}\ .
\end{aligned}$$

It follows that by (64) to (69), we have

$$\tau[\![C]\!] \quad = \quad \tau_{\text{tt}}[\![C]\!] \cup \tau_{\text{ff}}[\![C]\!]$$

where

$$\begin{aligned}\tau_{\text{tt}}[\![C]\!] &\triangleq \tau^B \cup \tau[\![S_t]\!] \cup \tau^t, \\ \tau_{\text{ff}}[\![C]\!] &\triangleq \tau^{\bar{B}} \cup \tau[\![S_f]\!] \cup \tau^f .\end{aligned}$$

By the conditions (59) and (78) on labelling of the conditional command $C$, we have $\tau_{\text{tt}}[\![C]\!] \circ \tau_{\text{ff}}[\![C]\!] = \tau_{\text{ff}}[\![C]\!] \circ \tau_{\text{tt}}[\![C]\!] = \emptyset$ so that

$$\tau^\star[\![C]\!] \quad = \quad (\tau_{\text{tt}}[\![C]\!])^\star \cup (\tau_{\text{ff}}[\![C]\!])^\star . \tag{79}$$

Intuitively the steps which are repeated in the conditional must all take place in one branch or the other since it is impossible to jump from one branch into the other.

Assume by induction hypothesis that

$$(\tau_{\text{tt}}[\![C]\!])^n \quad = \quad \tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n . \tag{80}$$

This holds for the basis $n = 1$ since $\tau[\![S_t]\!]^{-1} = \emptyset$ and $\tau[\![S_t]\!]^0 = 1_{\Sigma[\![P]\!]}$ is the identity. For $n \geq 1$, we have

$$\begin{aligned}
& (\tau_{\text{tt}}[\![C]\!])^{n+1} \\
= \quad & \wr \text{def. } t^{n+1} = t^n \circ t \int \\
& (\tau_{\text{tt}}[\![C]\!])^n \circ \tau_{\text{tt}}[\![C]\!] \\
= \quad & \wr \text{induction hypothesis} \int \\
& (\tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n) \circ \tau_{\text{tt}}[\![C]\!] \\
= \quad & \wr \circ \text{ distributes over } \cup \text{ (and } \circ \text{ has priority over } \cup) \int \\
& \tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \circ \tau_{\text{tt}}[\![C]\!] \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau_{\text{tt}}[\![C]\!] \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \circ \tau_{\text{tt}}[\![C]\!] \cup \\
& \tau[\![S_t]\!]^n \circ \tau_{\text{tt}}[\![C]\!] \\
= \quad & \wr \text{by the labelling scheme (59), (78) and the def. (64) to (69) of the possible transitions} \\
& \quad \text{so that } \tau^t \circ \tau_{\text{tt}}[\![C]\!] = \emptyset, \text{ etc.} \int \\
& \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau_{\text{tt}}[\![C]\!] \cup \tau[\![S_t]\!]^n \circ \tau_{\text{tt}}[\![C]\!] \\
= \quad & \wr \text{def. of } \tau_{\text{tt}}[\![C]\!] \text{ and } \circ \text{ distributes over } \cup \int \\
& \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau^B \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau[\![S_t]\!] \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n \circ \tau^B \cup \\
& \tau[\![S_t]\!]^n \circ \tau[\![S_t]\!] \cup \tau[\![S_t]\!]^n \circ \tau^t \\
= \quad & \wr \text{by the labelling scheme (59), (78) and the def. (64) to (69) of the possible transitions} \\
& \quad \text{so that } \tau^B \circ \tau^B = \emptyset,\ \tau[\![S_t]\!]^n \circ \tau^B, \text{ etc.} \int \\
& \tau^B \circ \tau[\![S_t]\!]^n \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^{n+1} \cup \tau[\![S_t]\!]^n \circ \tau^t \\
= \quad & \wr \cup \text{ is associative and commutative and def. (80) of } (\tau_{\text{tt}}[\![C]\!])^{n+1} \int \\
& (\tau_{\text{tt}}[\![C]\!])^{n+1} .
\end{aligned}$$

By recurrence, (80) holds for all $n \geq 1$ so that

$$\begin{aligned}
& (\tau_{\text{tt}}[\![C]\!])^\star \\
= \quad & \wr \text{def. } t^\star \int \\
& (\tau_{\text{tt}}[\![C]\!])^0 \cup \bigcup_{n\geq 1} (\tau_{\text{tt}}[\![C]\!])^n \\
= \quad & \wr \text{def. } t^0 \text{ and (80)} \int \\
& 1_{\Sigma[\![P]\!]} \cup \bigcup_{n\geq 1} (\tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n)
\end{aligned}$$

$$
\begin{aligned}
= \quad & \wr \circ \text{ distributes over } \cup \wr \\
& 1_{\Sigma[\![P]\!]} \cup \tau^B \circ \left( \bigcup_{n \geq 1} \tau[\![S_t]\!]^{n-2} \right) \circ \tau^t \cup \tau^B \circ \left( \bigcup_{n \geq 1} \tau[\![S_t]\!]^{n-1} \right) \cup \left( \bigcup_{n \geq 1} \tau[\![S_t]\!]^{n-1} \right) \circ \tau^t \cup \bigcup_{n \geq 1} \tau[\![S_t]\!]^n \\
= \quad & \wr \text{changing variables } k = n-2 \text{ and } j = n-1,\ \tau[\![S_t]\!]^{-1} = \emptyset,\ \tau[\![S_t]\!]^0 = 1_{\Sigma[\![P]\!]} \text{ and by the labelling scheme (59), (78) and the def. (64) to (69) of the possible transitions, } \tau^B \circ \tau^t = \emptyset, \text{ etc.} \wr \\
& \tau^B \circ \left( \bigcup_{k \geq 1} \tau[\![S_t]\!]^k \right) \circ \tau^t \cup \tau^B \circ \left( \bigcup_{j \geq 0} \tau[\![S_t]\!]^j \right) \cup \left( \bigcup_{j \geq 0} \tau[\![S_t]\!]^j \right) \circ \tau^t \cup \bigcup_{n \geq 0} \tau[\![S_t]\!]^n \\
= \quad & \wr \tau^B \circ \tau^t = \emptyset \text{ and def. of } t^\star \wr \\
& \tau^B \circ (\tau[\![S_t]\!])^\star \circ \tau^t \cup \tau^B \circ (\tau[\![S_t]\!])^\star \cup (\tau[\![S_t]\!])^\star \circ \tau^t \cup (\tau[\![S_t]\!])^\star \\
= \quad & \wr \circ \text{ distributes over } \cup \text{ (and } \star \text{ has priority over } \circ \text{ which has priority over } \cup) \wr \\
& (1_{\Sigma[\![P]\!]} \cup \tau^B) \circ (\tau[\![S_t]\!])^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \ .
\end{aligned}
$$

A similar result is easily established for $(\tau_{\text{ff}}[\![C]\!])^\star$ whence by (79), we get

$$
\begin{aligned}
\tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] \quad = \quad & (1_{\Sigma[\![P]\!]} \cup \tau^B) \circ (\tau[\![S_t]\!])^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup \\
& (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ (\tau[\![S_f]\!])^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^f) \ .
\end{aligned}
$$

3 —— The case of iteration is rather long to handle and can be skipped at first reading. By analogy with the conditional, the big-step operational semantics (94) of iteration should be intuitive. Formally, for the iteration $C = \texttt{while } B \texttt{ do } S \texttt{ od}$, we define

$$
\begin{aligned}
\tau^B &\triangleq \{\langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Mapsto \mathrm{tt}\}, \\
\tau^{\bar{B}} &\triangleq \{\langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Mapsto \mathrm{tt}\}, \\
\tau^R &\triangleq \{\langle\langle \mathrm{after}_P[\![S]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\} \ .
\end{aligned}
$$

It follows that by (70) to (73), we have

$$
\tau[\![C]\!] \quad = \quad \tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}} \ . \tag{81}
$$

We define the composition $\bigcirc_{i=1}^n t_i$ of relations $t_1, \ldots, t_n$ ($\circ$ is associative but not commutative so that the index set must be totally ordered for the notation to be meaningful):

$$
\begin{aligned}
\bigcirc_{i=1}^{n} t_i &\triangleq \emptyset, && \text{when} \quad n < 0, \\
\bigcirc_{i=1}^{0} t_i &\triangleq 1_{\Sigma[\![P]\!]}, && \text{when} \quad n = 0, \\
\bigcirc_{i=1}^{n} t_i &\triangleq t_1 \circ \ldots \circ t_n, && \text{when} \quad n > 0 \ .
\end{aligned}
$$

In order to compute $\tau^\star[\![C]\!] = \bigcup_{n \geq 0} \tau[\![C]\!]^n$ for the component $C = \texttt{while } B \texttt{ do } S \texttt{ od}$ of program $P$, we first compute the $n$-th power $\tau[\![C]\!]^n$ for $n \geq 0$. By recurrence $\tau[\![C]\!]^0 = 1_{\Sigma[\![P]\!]}$, $\tau[\![C]\!]^1 = \tau[\![C]\!] = \tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}$. For $n > 1$, we have

$$
\begin{aligned}
& (\tau[\![C]\!])^2 \\
= \quad & \wr \text{def. } t^2 = t \circ t \wr \\
& \tau[\![C]\!] \circ \tau[\![C]\!]
\end{aligned}
$$

$$
\begin{array}{ll}
= & \wr \text{def. (81) of } \tau[\![C]\!] \int \\
& (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \wr \circ \text{ distributes over } \cup \text{ (and } \circ \text{ has priority over } \cup) \int \\
& \tau^B \circ \tau^B \cup \tau[\![S]\!] \circ \tau^B \cup \tau^R \circ \tau^B \cup \tau^{\bar{B}} \circ \tau^B \cup \tau^B \circ \tau[\![S]\!] \cup \tau[\![S]\!] \circ \tau[\![S]\!] \cup \tau^R \circ \tau[\![S]\!] \cup \\
& \tau^{\bar{B}} \circ \tau[\![S]\!] \cup \tau^B \circ \tau^R \cup \tau[\![S]\!] \circ \tau^R \cup \tau^R \circ \tau^R \cup \tau^{\bar{B}} \circ \tau^R \cup \tau^B \circ \tau^{\bar{B}} \cup \tau[\![S]\!] \circ \tau^{\bar{B}} \cup \\
& \tau^R \circ \tau^{\bar{B}} \cup \tau^{\bar{B}} \circ \tau^{\bar{B}} \\
= & \wr \tau^B \circ \tau^B = \emptyset, \text{ by (71) and (56);} \\
& \quad \tau[\![S]\!] \circ \tau^B = \emptyset, \text{ by (72), (71), (78) and (60);} \\
& \quad \tau^{\bar{B}} \circ \tau^B = \emptyset, \text{ by (70), (71) and (56);} \\
& \quad \tau^R \circ \tau[\![S]\!] = \emptyset, \text{ by (73), (72) and (60);} \\
& \quad \tau^{\bar{B}} \circ \tau[\![S]\!] = \emptyset, \text{ by (70), (72), (78) and (60);} \\
& \quad \tau^B \circ \tau^R = \emptyset, \text{ by (71), (73) and (56);} \\
& \quad \tau^R \circ \tau^R = \emptyset, \text{ by (73), (60) and (78);} \\
& \quad \tau^B \circ \tau^{\bar{B}} = \emptyset, \text{ by (71), (70), (60) and (78)} \\
& \quad \tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset, \text{ by (72), (70), (60) and (78);} \\
& \quad \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset, \text{ by (70) and (56)} \int \\
& \tau^R \circ \tau^B \cup \tau^B \circ \tau[\![S]\!] \cup \tau[\![S]\!]^2 \cup \tau[\![S]\!] \circ \tau^R \cup \tau^R \circ \tau^{\bar{B}} \ .
\end{array}
$$

The generalization after computing the first few iterates $n = 1, \ldots, 4$ leads to the following induction hypothesis ($n \geq 1$)

$$
(\tau[\![C]\!])^n \quad \triangleq \quad A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup G_n \tag{82}
$$

where

$$
A_n \quad \triangleq \quad \bigcup_{n = \sum_{i=1}^{j} (k_i + 2)} \quad \bigodot_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \ ; \tag{83}
$$

(This corresponds to $j$ loops iterations from and to the loop entry at$_P[\![C]\!]$ where the $i$-th execution of the loop body $S$ exactly takes $k_i \geq 1$ [7] steps. $A_n = \emptyset$, $n \leq 1$.)

$$
B_n \quad \triangleq \quad \bigcup_{n = (\sum_{i=1}^{j} (k_i + 2)) + 1 + \ell} \left( \left( \bigodot_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau[\![S]\!]^{\ell} \right) \ ; \tag{84}
$$

(This corresponds to $j$ loops iterations from and to the loop entry at$_P[\![C]\!]$ where the $i$-th execution of the loop body $S$ exactly takes $k_i \geq 1$ steps followed by a successful condition $B$ and a partial execution of the loop body $S$ for $\ell \geq 0$ [8] steps. $B_0 = \emptyset$, $B_1 = \tau^B$.)

$$
C_n \quad \triangleq \quad \bigcup_{n = (\sum_{i=1}^{j} (k_i + 2)) + 1} \left( \left( \bigodot_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^{\bar{B}} \right) \ ; \tag{85}
$$

(This corresponds to $j$ loops iterations where the $i$-th execution of the loop body $S$ has $k_i \geq 1$ steps within $S$ until termination with condition $B$ false. $C_0 = \emptyset$, $C_1 = \tau^{\bar{B}}$.)

$$
D_n \quad \triangleq \quad \bigcup_{n = \ell + 1 + (\sum_{i=1}^{j} (k_i + 2))} \left( \tau[\![S]\!]^{\ell} \circ \tau^R \circ \left( \bigodot_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \right) \ ; \tag{86}
$$

[7] For short, the constraints $k_i > 0$, $i = 1, \ldots, j$ are not explicitly inserted in the formula.

[8] Again, the constraint $\ell \geq 0$ is left implicit in the formula.

(This corresponds to an observation of the execution starting in the middle of the loop body $S$ for $\ell$ steps followed by the jump back to the loop entry $\mathrm{at}_P[\![C]\!]$, followed by $j$ complete loops iterations from and to the loop entry $\mathrm{at}_P[\![C]\!]$ where the $i$-th execution of the loop body $S$ exactly takes $k_i \geq 1$ steps. $D_0 = \emptyset$, $D_1 = \tau^R$.)

$$E_n \triangleq \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+\ell+2+m} \left(\tau[\![S]\!]^\ell \circ \tau^R \circ \left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right) \circ \tau^B \circ \tau[\![S]\!]^m\right) ; \quad (87)$$

(This corresponds to an observation of the execution starting in the middle of the loop body $S$ for $\ell \geq 0$ steps followed by the jump back to the loop entry $\mathrm{at}_P[\![C]\!]$. Then there are $j$ loops iterations from and to the loop entry $\mathrm{at}_P[\![C]\!]$ where the $i$-th execution of the loop body $S$ exactly takes $k_i \geq 1$ steps. Finally the condition $B$ holds and a partial execution of the loop body $S$ for $m \geq 0$ steps is performed. $E_0 = E_1 = \emptyset$ and $E_2 = \tau^R \circ \tau^B$.)

$$F_n \triangleq \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+\ell+2} \left(\tau[\![S]\!]^\ell \circ \tau^R \circ \left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right) \circ \tau^{\bar{B}}\right) ; \quad (88)$$

(This case is similar to $E_n$ except that the execution of the loop terminates with condition $B$ false. $F_0 = F_1 = \emptyset$ and $F_2 = \tau^R \circ \tau^{\bar{B}}$.)

$$G_n \triangleq (\tau[\![S]\!])^n ; \quad (89)$$

(This case corresponds to the observation of $n \geq 1$ steps within the loop body $S$.).

We now proof (82) by recurrence on $n$. Given a formula $\mathcal{F}_n \in \{A_n, \ldots, F_n\}$ of the form $\mathcal{F}_n = \bigcup_{C(n,\ell,m,\ldots)} \mathcal{T}(n, \ell, m, \ldots)$, where $n, \ell, m, \ldots$ are free variables of the condition $C$ and term $\mathcal{T}$, we write $\mathcal{F}_n \mid C'(n, \ell, m, \ldots)$ for the formula $\bigcup_{C(n,\ell,m,\ldots)\wedge C'(n,\ell,m,\ldots)} \mathcal{T}(n, \ell, m, \ldots)$.

3.1 — For the basis observe that for $n = 1$, $A_1 = \emptyset$, $B_1 = \tau^B$, $C_1 = \tau^{\bar{B}}$, $D_1 = \tau^R$, $E_1 = \emptyset$, $F_1 = \emptyset$ and $G_1 = (\tau[\![S]\!])^1 = \tau[\![S]\!]$ so that

$$\begin{aligned}
(\tau[\![C]\!])^1 &= \tau[\![C]\!] \\
&= \tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}} \\
&= B_1 \cup G_1 \cup D_1 \cup C_1 \\
&= A_1 \cup B_1 \cup C_1 \cup D_1 \cup E_1 \cup F_1 \cup G_1 .
\end{aligned}$$

3.2 — For $n = 2$, observe that $A_2 = \emptyset$, $B_2 = \tau^B \circ \tau[\![S]\!]$, $C_2 = \emptyset$, $D_2 = \tau[\![S]\!] \circ \tau^R$, $E_2 = \tau^R \circ \tau^B$, $F_2 = \tau^R \circ \tau^{\bar{B}}$ and $G_2 = (\tau[\![S]\!])^2$ so that

$$\begin{aligned}
(\tau[\![C]\!])^2 &= \tau^R \circ \tau^B \cup \tau^B \circ \tau[\![S]\!] \cup \tau[\![S]\!]^2 \cup \tau[\![S]\!] \circ \tau^R \cup \tau^R \circ \tau^{\bar{B}} \\
&= E_2 \cup B_2 \cup G_2 \cup D_2 \cup E_2 \cup F_2 \\
&= A_2 \cup B_2 \cup C_2 \cup D_2 \cup E_2 \cup F_2 \cup G_2 .
\end{aligned}$$

3.3 — For the induction step $n \geq 2$, we have to consider the compositions $A_n \circ \tau[\![C]\!], \ldots, G_n \circ \tau[\![C]\!]$ in turn.

— $A_n \circ \tau[\![C]\!]$

$=$ ≀def. (81) of $\tau[\![C]\!]$≀

$A_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ ≀$\circ$ distributes over $\cup$, $n \geq 2$ so $j \geq 1$ whence $A_n = \tau' \circ \tau^R$, $\tau^R \circ \tau[\![S]\!] = \emptyset$ and $\tau^R \circ \tau^R = \emptyset$≀

$A_n \circ \tau^B \cup A_n \circ \tau^{\bar{B}}$

$=$ ≀def. (83) of $A_n$ and $\tau[\![S]\!]^0 = 1_{\Sigma[\![P]\!]}$≀

$$\left( \bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+1+0} \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau[\![S]\!]^0$$

$$\cup \left( \bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+1} \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^{\bar{B}}$$

$=$ ≀def. (84) of $B_{n+1}$ with additional constraint $\ell = 0$ and def. (85) of $C_{n+1}$≀

$B_{n+1} \mid \ell = 0 \ \cup \ C_{n+1}$ .

— $B_n \circ \tau[\![C]\!]$

$=$ ≀def. (81) of $\tau[\![C]\!]$≀

$B_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ ≀$\circ$ distributes over $\cup$, either $\ell = 0$ in $B_n$, in which case $B_n = \tau' \circ \tau^B$, $\tau^B \circ \tau^B = \emptyset$, $\tau^B \circ \tau^R = \emptyset$ and $\tau^B \circ \tau^{\bar{B}} = \emptyset$ or $\ell > 0$ in $B_n$, in which case $B_n = \tau'' \circ \tau[\![S]\!]$, $\tau[\![S]\!] \circ \tau^B = \emptyset$ and $\tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset$≀

$(B_n \mid \ell = 0) \circ \tau[\![S]\!] \cup (B_n \mid \ell > 0) \circ \tau[\![S]\!] \cup (B_n \mid \ell > 0) \circ \tau^R$

$=$ ≀def. (84) of $B_n$≀

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup$

$$\left( \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+1+\ell} \left( \left( \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau[\![S]\!]^{\ell} \right) \right) \circ \tau^R$$

$=$ ≀$\circ$ distributes over $\cup$≀

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup$

$$\bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+2+\ell} \left( \left( \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ (\tau^B \circ \tau[\![S]\!]^{\ell} \circ \tau^R) \right)$$

$=$ ≀by letting $k_{j+1} = \ell \geq 1$≀

$$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup \bigcup_{n+1=\sum_{i=1}^{j+1}(k_i+2)} \left( \bigcirc_{i=1}^{j+1} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right)$$

$=$ ≀by letting $j' = j + 1$ and def. (84) of $A_{n+1}$≀

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup A_{n+1}$

$=$ ≀associativity of $\cup$≀

$(B_{n+1} \mid \ell > 0) \cup A_{n+1}$ .

— $C_n \circ \tau[\![C]\!]$

$=$ ≀def. (81) of $\tau[\![C]\!]$≀

$$
\begin{array}{ll}
 & C_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup,\ C_n = \tau' \circ \tau^{\bar{B}} \text{ and } \tau^{\bar{B}} \circ \tau^B = \tau^{\bar{B}} \circ \tau[\![S]\!] = \tau^{\bar{B}} \circ \tau^R = \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset \wr \\
 & \emptyset\ .
\end{array}
$$

$$
\begin{array}{ll}
\text{—} & D_n \circ \tau[\![C]\!] \\
= & \quad \wr \text{def. (81) of } \tau[\![C]\!] \wr \\
 & D_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup,\ D_n \text{ has the form } \tau' \circ \tau^R \text{ and } \tau^R \circ \tau[\![S]\!] = \tau^R \circ \tau^R = \emptyset \wr \\
 & D_n \circ \tau^B \cup D_n \circ \tau^{\bar{B}} \\
= & \quad \wr \text{def. (87) of } E_n \text{ and (88) of } F_n \wr \\
 & (E_{n+1} \mid m = 0) \cup F_{n+1}\ .
\end{array}
$$

$$
\begin{array}{ll}
\text{—} & E_n \circ \tau[\![C]\!] \\
= & \quad \wr \text{def. (81) of } \tau[\![C]\!] \wr \\
 & E_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup,\ E_n \mid m = 0 \text{ has the form } \tau' \circ \tau^B \text{ while } E_n \mid m >= 0 \text{ has the} \\
 & \quad \text{form } \tau'' \circ \tau[\![S]\!],\ \tau^B \circ \tau^B = \tau^B \circ \tau^R = \tau^B \circ \tau^{\bar{B}} = \emptyset \text{ and } \tau[\![S]\!] \circ \tau^B = \tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset \wr \\
 & (E_n \mid m = 0) \circ \tau[\![S]\!] \cup (E_n \mid m > 0) \circ \tau[\![S]\!] \cup (E_n \mid m > 0) \circ \tau^R \\
= & \quad \wr \text{def. (87) of } E_n \text{ and (86) of } D_{n+1} \text{ where } k_i = m \geq 1 \text{ so that } \ell < n \wr \\
 & (E_{n+1} \mid m = 1) \cup (E_{n+1} \mid m > 1) \cup (D_{n+1} \mid \ell < n) \\
= & \quad \wr \cup \text{ is associative} \wr \\
 & (E_{n+1} \mid m > 0) \cup (D_{n+1} \mid \ell < n)\ .
\end{array}
$$

$$
\begin{array}{ll}
\text{—} & F_n \circ \tau[\![C]\!] \\
= & \quad \wr \text{def. (81) of } \tau[\![C]\!] \wr \\
 & F_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup, \text{ by def. (88) of } F_n \text{ has the form } \tau' \circ \tau^{\bar{B}} \text{ and } \tau^{\bar{B}} \circ \tau^B = \tau^{\bar{B}} \circ \tau[\![S]\!] \\
 & \quad = \tau^{\bar{B}} \circ \tau^R = \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset \wr \\
 & \emptyset\ .
\end{array}
$$

$$
\begin{array}{ll}
\text{—} & G_n \circ \tau[\![C]\!] \\
= & \quad \wr \text{def. (89) of } G_n \text{ and (81) of } \tau[\![C]\!] \wr \\
 & (\tau[\![S]\!])^n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup,\ n \geq 1,\ \tau[\![S]\!] \circ \tau^B = \tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset \wr \\
 & (\tau[\![S]\!])^n \circ \tau[\![S]\!] \cup (\tau[\![S]\!])^n \circ \tau^R) \\
= & \quad \wr \text{def. } n+1\text{-th power and (86) of } D_{n+1} \wr \\
 & (\tau[\![S]\!])^{n+1} \cup (D_{n+1} \mid \ell = n)\ .
\end{array}
$$

Grouping all cases together, we get

$$
\begin{array}{ll}
 & (\tau[\![C]\!])^{n+1} \\
= & \quad \wr \text{def. } n+1\text{-th power and (82)} \wr \\
 & (A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup G_n) \circ (\tau[\![C]\!])^n \\
= & \quad \wr \circ \text{ distributes over } \cup, \text{ def. (89) of } G_n \wr \\
 & (A_n \circ \tau[\![C]\!] \cup B_n \circ \tau[\![C]\!] \cup C_n \circ \tau[\![C]\!] \cup D_n \circ \tau[\![C]\!] \cup E_n \circ \tau[\![C]\!] \cup F_n \circ \tau[\![C]\!] \circ (\tau[\![C]\!])^n \circ \tau[\![C]\!] \\
= & \quad \wr \text{replacing according to the above lemmata} \wr \\
 & (B_{n+1} \mid \ell = 0 \cup C_{n+1}) \cup ((B_{n+1} \mid \ell > 0) \cup A_{n+1}) \cup \emptyset \cup ((E_{n+1} \mid m = 0) \cup F_{n+1}) \cup ((E_{n+1} \mid \\
 & m > 0) \cup (D_{n+1} \mid \ell < n)) \cup \emptyset \cup ((\tau[\![S]\!])^{n+1} \cup (D_{n+1} \mid \ell = n)) \\
= & \quad \wr \cup \text{ is associative and commutative and } (D_{n+1} \mid \ell > n) = \emptyset \wr \\
 & A_{n+1} \cup B_{n+1} \cup C_{n+1} \cup D_{n+1} \cup E_{n+1} \cup F_{n+1} \cup G_{n+1}
\end{array}
$$

By recurrence on $n \geq 1$, we have proved that

$$(\tau[\![C]\!])^n \quad \stackrel{\Delta}{=} \quad A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup (\tau[\![C]\!])^n$$

so that

$$\begin{array}{ll} & \tau^\star[\![C]\!] \\ = & (\tau[\![C]\!])^\star \\ = & (\tau[\![C]\!])^0 \cup \bigcup_{n\geq 1}(A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup (\tau[\![S]\!])^n) \\ = & (\tau[\![C]\!])^0 \cup (\bigcup_{n\geq 1} A_n \cup \bigcup_{n\geq 1} B_n \cup \bigcup_{n\geq 1} C_n \cup \bigcup_{n\geq 1} D_n \cup \bigcup_{n\geq 1} E_n \cup \bigcup_{n\geq 1} F_n \cup (\tau[\![S]\!])^\star) \ . \end{array}$$

We now compute each of these terms.

$$\begin{array}{ll} & \bigcup_{n\geq 1} A_n \\ = & \quad \wr \text{def. (83) of } A_n \wr \\ & \bigcup_{n\geq 1} \quad \bigcup_{n=\sum_{i=1}^{j}(k_i+2)} \quad \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \\ = & \quad \wr \text{for } n \in [1,3] \text{ this is } \emptyset \text{ while for } n > 3 \text{, we can always find } j \text{ and } k_1 \geq 1, ..., k_j \geq 1 \\ & \quad \text{such that } n = \sum_{i=1}^{j}(k_i+2). \text{ Reciprocally, for all choices of j and } k_1 \geq 1, ..., k_j \geq 1 \\ & \quad \text{there exists an } n > 3 \text{ such that } n = \sum_{i=1}^{j}(k_i+2). \wr \\ & (\tau^B \circ \tau[\![S]\!]^+ \circ \tau^R)^+ \\ = & \quad \wr \tau^B \circ \tau^R = \emptyset \wr \\ & (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^+ \ . \end{array}$$

By the same reasoning, we get

$$\begin{array}{rcl} \bigcup_{n\geq 1} B_n & = & (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^B \circ \tau[\![S]\!]^\star, \\ \bigcup_{n\geq 1} C_n & = & (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^{\bar{B}}, \\ \bigcup_{n\geq 1} D_n & = & \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star, \\ \bigcup_{n\geq 1} E_n & = & \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^B \circ \tau[\![S]\!]^\star, \\ \bigcup_{n\geq 1} F_n & = & \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} \ . \end{array}$$

Grouping now all cases together and using the fact that $\circ$ distributes over $\cup$, we finally get

$$\begin{array}{rcl} \tau^\star[\![C]\!] & = & \tau[\![S]\!]^0 \cup (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^+ \cup (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^B \circ \tau[\![S]\!]^\star \\ & & \quad \cup (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} \cup \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \\ & & \quad \cup \ \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^B \circ \tau[\![S]\!]^\star \\ & & \quad \cup \ \tau[\![S]\!]^\star \circ \tau^R \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} \\ & = & (1_{\Sigma[\![P]\!]} \cup \tau[\![S]\!]^\star \circ \tau^R) \circ (\tau^B \circ \tau[\![S]\!]^\star \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau[\![S]\!]^\star \cup \tau^{\bar{B}}) \ . \end{array}$$

4 — The case of sequence is also long to handle and can be skipped at first reading. The big-step operational semantics (95) of the sequence is indeed rather intuitive. Formally, for the sequence $S = C_1$ ; ... ; $C_n$, $n \geq 1$, we first prove a lemma.

4.1 — Let $P$ be the program with subcommand $S = C_1$ ; ... ; $C_n$. Successive small steps in $S$ must be made in sequence since, by the definition (76) and (74) of $\tau[\![S]\!]$ and the labelling scheme (58), it is impossible to jump from one command into a different one

$$\begin{aligned}
&\tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \quad = \qquad\qquad (90)\\
&\quad (\forall i \in [1, n] : k_i = 0 \ \textbf{?}\ 1_{\Sigma[\![P]\!]}\\
&\quad \textbf{|}\ \exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j]) \ \textbf{?}\ \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \ \textbf{¿}\ \emptyset) \ .
\end{aligned}$$

The proof is by recurrence on $n$.

4.1.1 — If, for the basis, $n = 1$ then either $k_1 = 0$ and $\tau^0[\![C_1]\!] = 1_{\Sigma[\![P]\!]}$ or $k_1 > 0$ and then $\tau^{k_1}[\![C_1]\!] = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!]$ by choosing $i = j = 1$.

4.1.2 — For the induction step, assuming (90), we prove that

$$T \quad = \quad \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!]$$

is of the form (90) with $n + 1$ substituted for $n$. Two cases, with several subcases have to be considered.

4.1.2.1 — If $\forall i \in [1, n] : k_i = 0$ then we consider two subcases.

4.1.2.1.1 — If $k_{n+1} = 0$ then $\forall i \in [1, n + 1] : k_i = 0$ and $T = \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]} \circ \tau^0[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]}$.

4.1.2.1.2 — Otherwise $k_{n+1} > 0$ and then $\forall \ell \in [1, n+1] : (k_\ell \neq 0 \iff \ell \in [n+1, n+1])$ and $T = \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]} \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!]$ by choosing $i = j = n + 1$.

4.1.2.2 — Otherwise, $\exists i \in [1, n] : k_i \neq 0$.

4.1.2.2.1 — If $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j])$ then by (90), we have

$$T \quad = \quad \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] \ .$$

4.1.2.2.1.1 – If $k_{n+1} = 0$ then $\exists 1 \leq i \leq j \leq n + 1 : \forall \ell \in [1, n + 1] : (k_\ell \neq 0 \iff \ell \in [i, j])$ and:

$$\begin{aligned}
T \quad &= \quad \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!],\\
&= \quad \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ 1_{\Sigma[\![P]\!]},\\
&= \quad \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \ .
\end{aligned}$$

4.1.2.2.1.2 – Otherwise $k_{n+1} > 0$ and we distinguish two subcases.

4.1.2.2.1.2.1 - If $j < n$ then $t^{k+1} = t \circ t^k = t^k \circ t$ so

$$T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j - 1}[\![C_j]\!] \circ \circ \tau[\![C_j]\!] \circ \tau[\![C_{n+1}]\!] \circ \tau^{k_{n+1}-1}[\![C_{n+1}]\!] .$$

By the definition (76) and (74) of $\tau[\![C]\!]$ and the labelling scheme (58), we have $\tau[\![C_j]\!] \circ \tau[\![C_{n+1}]\!] = \emptyset$ since $j < n$ so that in that case $T = \emptyset$.

4.1.2.2.1.2.2 - Otherwise $j = n$ so $\forall \ell \in [0, i[: k_\ell = 0$, $\forall \ell \in [i, n+1] : k_\ell > 0$ and $T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!]$ whence $\forall \ell \in [1, n+1] : (k_\ell \neq 0 \iff \ell \in [i, j])$ with $1 \leq i < j = n+1$ and $T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!]$.

4.1.2.2.2 — Otherwise $\forall 1 \leq i \leq j \leq n : \exists \ell \in [1, n] : (k_\ell \neq 0 \wedge \ell \notin [i, j]) \vee (\ell \in [i, j] \wedge k_\ell = 0)$.

4.1.2.2.2.1 – This excludes $n = 1$ since then $i = j = \ell = 1$ and $k_1 = 0$ in contradiction with $\exists i \in [1, n] : k_i \neq 0$.

4.1.2.2.2.2 – If $n = 2$ then $k_1 = 0$ and $k_2 > 0$ or $k_1 > 0$ and $k_2 = 0$ which corresponds to case 4.1.2.2.1, whence is impossible.

4.1.2.2.2.3 – So necessarily $n \geq 3$. Let $p \in [1, n]$ be minimal and $q \in [1, n]$ be maximal such that $k_p \neq 0$ ad $k_q \neq 0$. There exists $m \in [p, q]$ such that $k_m = 0$ since otherwise $k_\ell \neq 0$ and either $\ell < p$ in contradiction with the minimality of$p$ or $\ell > j$ in contradiction with the maximality of $q$. We have $p < m < q$ with $k_p \neq 0$, $k_m = 0$ and $k_j = 0$. Assume $m$ to be minimal with that property, so that $k_{m-1} \neq 0$ and then that $q'$ is the minimal $q$ with this property so that $k_{q'-1} = 0$. We have $k_1 = 0, \ldots, k_{p-1} = 0, k_p \neq 0, \ldots, k_{m-1} = 0, k_m = 0$, $k_{q'-1} = 0$ $k_{q'} \neq 0, \ldots$ It follows, by the definition (76) and (74) of $\tau[\![C]\!]$ and the labelling scheme (58) that $\tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] = \emptyset$ that $T = \emptyset \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = \emptyset$.

It remains to prove that

$$\forall 1 \leq i \leq j \leq n+1 : \exists \ell \in [1, n+1] : (k_\ell \neq 0 \wedge \ell \notin [i, j]) \vee (\ell \in [i, j] \wedge k_\ell = 0) .$$

4.1.2.2.2.3.1 - If $j < n+1$ then this follows from (90).

4.1.2.2.2.3.2 - Otherwise $j = n+1$ in which case either $k_{n+1} = 0$ and then we choose $\ell = j$ or $k_{n+1} > 0$ so that $q' = j = n+1$. If $j \leq m$ then for $\ell = m$, we have $k_\ell = k_m = 0$. Otherwise $m < i \leq q'$. Choosing $\ell = p$, we have $\ell \in [1, j]$ with $k_\ell = k_p \neq 0$.

4.2 — We will need a second lemma, stating that $k$ small steps in $C_1$ ; … ; $C_n$ must be made in sequence with $k_1$ steps in $C_1$, followed by $k_2$ in $C_2$, … , followed by $k_n$ in $C_n$ such that the total number $k_1 + \ldots + k_n$ of these steps is precisely $k$

$$\tau^k[\![C_1 \;;\; \ldots \;;\; C_n]\!] = \bigcup_{k = k_1 + \ldots + k_n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] . \tag{91}$$

The proof is by recurrence on $k \geq 0$.

4.2.1 — For $k = 0$, we get $k_1 = \ldots = k_n = 0$ and $1_{\Sigma[\![P]\!]}$ on both sides of the equality.

4.2.2 — For $k = 1$, there must exist $m \in [1, n]$ such that $k_m = 1$ while for all $j \in [1, n] - \{m\}$, $k_j = 0$. By the definition (76) and (74) of $\tau[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]$, we have

$$\tau[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \quad = \quad \bigcup_{m=1}^{n} \tau[\![C_m]\!] \ .$$

4.2.3 — For the induction step $k \geq 2$, we have

$$\begin{array}{ll}
& \tau^{k+1}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \\
= & \quad \wr\text{def. } t^{k+1} = t^k \circ t \text{ of powers}\wr \\
& \tau^{k}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \circ \tau[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \\
= & \quad \wr\text{def. (76) and (74) of } \tau[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]\wr \\
& \tau^{k}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \circ \displaystyle\bigcup_{m=1}^{n} \tau[\![C_m]\!] \\
= & \quad \wr\circ \text{ distributes over } \cup\wr \\
& \displaystyle\bigcup_{m=1}^{n} \tau^{k}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \circ \tau[\![C_m]\!] \\
= & \quad \wr\text{induction hypothesis (91)}\wr \\
& \displaystyle\bigcup_{m=1}^{n} \left( \bigcup_{k=k_1+\ldots+k_n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \right) \circ \tau[\![C_m]\!] \\
= & \quad \wr\circ \text{ distributes over } \cup\wr \\
& \displaystyle\bigcup_{k=k_1+\ldots+k_n} \bigcup_{m=1}^{n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau[\![C_m]\!] \\
\triangleq & \quad \wr\text{by definition}\wr \\
& T \ .
\end{array}$$

4.2.3.1 — We first show that

$$T \quad \subseteq \quad \bigcup_{k+1=k'_1+\ldots+k'_n} \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \ .$$

According to lemma (90), three cases have to be considered for

$$t \quad \triangleq \quad \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau[\![C_m]\!] \ .$$

4.2.3.1.1 – The case $\forall i \in [1, n] : k_i = 0$ is impossible since then $k = \sum_{j=1}^{n} k_j = 0$ in contradiction with $k \geq 2$.

4.2.3.1.2 – Else if $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j])$ then

$$t \quad \triangleq \quad \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau[\![C_m]\!] \ .$$

We discriminate according to the value of $m$.

4.2.3.1.2.1 – If $m = j$, we get

$$\begin{array}{rcl} t & = & \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j+1}[\![C_j]\!], \\ & = & \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \end{array}$$

with $k+1 = k'_1 + \ldots + k'_n$ where $k'_1 = 0, \ldots, k'_{i-1} = 0, k'_i = k_i, \ldots, k'_j = k_j + 1, k'_{j+1} = 0, \ldots, k'_n = 0$.

4.2.3.1.2.2 – If $m = j + 1$, we get

$$\begin{array}{rcl} t & = & \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^{1}[\![C_{j+1}]\!], \\ & = & \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \end{array}$$

with $k+1 = k'_1 + \ldots + k'_n$ where $k'_1 = 0, \ldots, k'_{i-1} = 0, k'_i = k_i, \ldots, k'_j = k_j, k'_{j+1} = 1, k'_{j+2} = 0, \ldots, k'_n = 0$.

4.2.3.1.2.3 – Otherwise, by the definition (76) and (74) of $\tau[\![C]\!]$ and the labelling scheme (58), $\tau[\![C_j]\!] \circ \tau[\![C_m]\!] = \emptyset$ so that $T = \emptyset$ that is $t = \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!]$ with $k'_\ell = k_\ell$ for $\ell \in [1, n] - \{m\}$ and $k'_m = k_m + 1$.

4.2.3.1.3 — Otherwise $T = \emptyset$ so that the inclusion is trivial.

4.2.3.2 — Inversely, we now show that

$$\bigcup_{k+1=k'_1+\ldots+k'_n} \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \quad \subseteq \quad T\ .$$

According to lemma (90), three cases have to be considered for

$$t \quad \triangleq \quad \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!]\ .$$

4.2.3.2.1 – If $\forall i \in [1, n] : k'_i = 0$ then $k + 1 = \sum_{i=1}^{n} k'_i = 0$ which is impossible with $k \geq 0$.

4.2.3.2.2 – Else if $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k'_\ell \neq 0 \iff \ell \in [i, j])$ then

$$t \quad \triangleq \quad \tau^{k'_i}[\![C_i]\!] \circ \ldots \circ \tau^{k'_j}[\![C_j]\!]\ .$$

with all $k'_i > 0, \ldots, k'_j > 0$.

4.2.3.2.2.1 – If $k'_j = 1$ then $t$ is

$$t \quad \triangleq \quad \tau^{k'_i}[\![C_i]\!] \circ \ldots \circ \tau^{k'_{j-1}}[\![C_{j-1}]\!] \circ \tau^{1}[\![C_j]\!]\ .$$

so we choose $k_1 = 0, \ldots, k_{i-1} = 0, k_i = k'_i, \ldots, k_{j-1} = k'_{j-1}, k_j = 0, \ldots, k_n = 0$ and $m = j$ with $k + 1 = k'_1 + \ldots + k'_n$ whence $k = k_1 + \ldots + k_n$.

4.2.3.2.2.2 – Otherwise $k'_j > 1$ then we have $t$ of the form required for $T$ by choosing $k_1 = 0, \ldots, k_{i-1} = 0, k_i = k'_i, \ldots, k_j = k'_j - 1, k_{j+1} = 0, \ldots, k_n = 0$ and $m = j$ with $k + 1 = k'_1 + \ldots + k'_n$ whence $k = k_1 + \ldots + k_n$.

4.2.3.2.3 — Otherwise $t = \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!]$ is $\emptyset$ which is obviously included in $T$.

4.3 — We can now consider the case 4 of the sequence $S = C_1 \;;\; \ldots \;;\; C_n$, $n \geq 1$

$$
\begin{array}{ll}
& \tau^\star[\![C_1 \;;\; \ldots \;;\; C_n]\!] \\
= & \quad \wr\text{def. reflexive transitive closure}\wr \\
& \bigcup_{k \geq 0} \tau^k[\![C_1 \;;\; \ldots \;;\; C_n]\!] \\
= & \quad \wr\text{lemma (91)}\wr \\
& \bigcup_{k_1 + \ldots + k_n \geq 0} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \\
= & \bigcup_{k_1 \geq 0} \ldots \bigcup_{k_n \geq 0} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \\
= & \quad \wr\circ\text{ distributes over } \cup\wr \\
& \left(\bigcup_{k_1 \geq 0} \tau^{k_1}[\![C_1]\!]\right) \circ \ldots \circ \left(\bigcup_{k_n \geq 0} \tau^{k_n}[\![C_n]\!]\right) \\
= & \quad \wr\text{def. reflexive transitive closure}\wr \\
& \tau^\star[\![C_1]\!] \circ \ldots \circ \tau^\star[\![C_n]\!] \;.
\end{array}
$$

5 — Finally, for programs $P = S \;;;$, we have

$$
\begin{array}{ll}
& \tau^\star[\![S \;;;]\!] \\
= & \quad \wr\text{def. reflexive transitive closure}\wr \\
& \bigcup_{k \geq 0} \tau^k[\![S \;;;]\!] \\
= & \quad \wr\text{by the definition (76) and (75) of } \tau[\![S \;;;]\!]\wr \\
& \bigcup_{k \geq 0} \tau^k[\![S]\!] \\
= & \quad \wr\text{def. reflexive transitive closure}\wr \\
& \tau^\star[\![S]\!] \;.
\end{array}
$$

In conclusion the calculational design of the reflexive transitive closure of the program transition relation leads to the functional and compositional characterization given in Fig. 13. Here compositional means, in the sense of denotational semantics, by induction on the program syntactic structure. Observe that contrary to the classical big step operational or natural semantics [31], the effect of execution is described not only from entry to exit states but also from any (intermediate) state to any subsequently reachable state. This is better adapted to our later reachability analysis.

### *12.9 Predicate transformers and fixpoints*

The *pre-image* $\text{pre}[t]\,P$ of a set $P \subseteq S$ of states by a transition relation $t \subseteq S \times S$ is the set of states from which it is possible to reach a state in $P$ by a transition $t$

$$\text{pre}[t]\,P \;\triangleq\; \{s \mid \exists s' : \langle s,\, s'\rangle \in t \wedge s' \in P\} \;.$$

The *dual pre-image* $\widetilde{\text{pre}}[t]\,P$ is the set of states from which any transition, if any, must lead to a state in $P$

$$\bullet \quad \tau^\star[\![\texttt{skip}]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{skip}]\!] \tag{92}$$

$$\bullet \quad \tau^\star[\![\mathsf{X} \texttt{ := } A]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\mathsf{X} \texttt{ := } A]\!]$$

$$\bullet \quad \tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] = (1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f) \tag{93}$$

where:

$$\tau^B \triangleq \{\langle\langle \mathrm{at}_P[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Mapsto \mathrm{tt}\}$$

$$\tau^{\bar{B}} \triangleq \{\langle\langle \mathrm{at}_P[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_f]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Mapsto \mathrm{tt}\}$$

$$\tau^t \triangleq \{\langle\langle \mathrm{after}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}$$

$$\tau^f \triangleq \{\langle\langle \mathrm{after}_P[\![S_f]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}$$

$$\bullet \quad \tau^\star[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!] = (1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}) \tag{94}$$

where:

$$\tau^B \triangleq \{\langle\langle \mathrm{at}_P[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Mapsto \mathrm{tt}\}$$

$$\tau^{\bar{B}} \triangleq \{\langle\langle \mathrm{at}_P[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Mapsto \mathrm{tt}\}$$

$$\tau^R \triangleq \{\langle\langle \mathrm{after}_P[\![S]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}$$

$$\bullet \quad \tau^\star[\![C_1 \texttt{ ; } \ldots \texttt{ ; } C_n]\!] = \tau^\star[\![C_1]\!] \circ \ldots \circ \tau^\star[\![C_n]\!] \tag{95}$$

$$\bullet \quad \tau^\star[\![S \texttt{ ;;}]\!] = \tau^\star[\![S]\!]\ . \tag{96}$$

Figure 13: Big step operational semantics

$$\widetilde{\mathrm{pre}}[t]\,P \triangleq \neg\,\mathrm{pre}[t](\neg P) = \{s \mid \forall s' : \langle s,\ s'\rangle \in t \implies s' \in P\}\ .$$

The *post-image* $\mathrm{post}[t]\,P$ is the inverse pre-image, that is the set of states which are reachable from $P \subseteq S$ by a transition $t$

$$\mathrm{post}[t]\,P \triangleq \mathrm{pre}[t^{-1}]\,P = \{s' \mid \exists s : s \in P \wedge \langle s,\ s'\rangle \in t\}\ . \tag{97}$$

The *dual post-image* $\widetilde{\mathrm{post}}[t]\,P$ is the set of states which can only be reached, if ever possible, by a transition $t$ from $P$

$$\widetilde{\mathrm{post}}[t]\,P \triangleq \neg\,\mathrm{post}[t](\neg P) = \{s' \mid \forall s : \langle s,\ s'\rangle \in t \implies s \in P\}\ .$$

We have the Galois connections ($t \subseteq S \times S$)

$$\langle \wp(S),\ \subseteq\rangle \underset{\mathrm{post}[t]}{\overset{\widetilde{\mathrm{pre}}[t]}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle, \qquad \langle \wp(S),\ \subseteq\rangle \underset{\mathrm{pre}[t]}{\overset{\widetilde{\mathrm{post}}[t]}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle$$

as well as ($P \subseteq S$, $\gamma_P(Y) \triangleq \{\langle s,\ s'\rangle \mid s \in P \implies s' \in Y\}$)

$$\langle \wp(S \times S),\ \subseteq\rangle \underset{\lambda t\bullet \mathrm{post}[t]\,P}{\overset{\gamma_P}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle, \quad \langle \wp(S \times S),\ \subseteq\rangle \underset{\lambda t\bullet \mathrm{pre}[t]\,P}{\overset{\gamma_P{}^{-1}}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle\ . \tag{98}$$

We often use the fact that

$$\begin{array}{rclcrcll} \mathrm{post}[t_1 \circ t_2] &=& \mathrm{post}[t_2] \circ \mathrm{post}[t_1], & & \mathrm{pre}[t_1 \circ t_2] &=& \mathrm{pre}[t_1] \circ \mathrm{pre}[t_2], & (99)\\ \mathrm{post}[1_S]\, P &=& P & \text{and} & \mathrm{pre}[1_S]\, P &=& P\ . & (100) \end{array}$$

The following fixpoint characterizations are classical (see e.g. [4], [5])

$$\begin{array}{rclcl} \mathrm{pre}[t^\star]\, F &=& \mathrm{lfp}^{\subseteq}\, \lambda X \centerdot F \cup \mathrm{pre}[t]\, X &=& \mathrm{lfp}^{\subseteq}_{F}\, \lambda X \centerdot X \cup \mathrm{pre}[t]\, X,\\ \widetilde{\mathrm{pre}}[t^\star]\, F &=& \mathrm{gfp}^{\subseteq}\, \lambda X \centerdot F \cap \widetilde{\mathrm{pre}}[t]\, X &=& \mathrm{gfp}^{\subseteq}_{F}\, \lambda X \centerdot X \cap \widetilde{\mathrm{pre}}[t]\, X,\\ \mathrm{post}[t^\star]\, I &=& \mathrm{lfp}^{\subseteq}\, \lambda X \centerdot I \cup \mathrm{post}[t]\, X &=& \mathrm{lfp}^{\subseteq}_{I}\, \lambda X \centerdot X \cup \mathrm{post}[t]\, X,\\ \widetilde{\mathrm{post}}[t^\star]\, I &=& \mathrm{gfp}^{\subseteq}\, \lambda X \centerdot I \cap \widetilde{\mathrm{post}}[t]\, X &=& \mathrm{gfp}^{\subseteq}_{I}\, \lambda X \centerdot X \cap \widetilde{\mathrm{post}}[t]\, X\ . \end{array} \tag{101}$$

### *12.10 Reachable states collecting semantics*

The reachable states collecting semantics of a component $C \in \mathrm{Cmp}[\![P]\!]$ of a program $P \in \mathrm{Prog}$ is the set $\mathrm{post}[\tau^\star[\![C]\!]](In)$ of states which are reachable from a given set $In \in \wp(\Sigma[\![P]\!])$ of initial states, in particular the entry states $In = \mathrm{Entry}[\![P]\!]$ when $C$ is the program $P$. The program analysis problem we are interested in is to effectively compute a machine representable program invariant $J \in \wp(\Sigma[\![P]\!])$ such that

$$\mathrm{post}[\tau^\star[\![C]\!]]\, In \;\subseteq\; J\ . \tag{102}$$

Using (101), the collecting semantics $\mathrm{post}[\tau^\star[\![C]\!]]\, In$ can be expressed in fixpoint form, as follows

$$\begin{array}{lrcll} & \mathrm{Post}[\![C]\!] &\in& \wp(\Sigma[\![P]\!]) \xrightarrow{\text{cjm}} \wp(\Sigma[\![P]\!]), &\\ & \mathrm{Post}[\![C]\!]In &\triangleq& \mathrm{post}[\tau^\star[\![C]\!]]\, In & (103)\\ & &=& \mathrm{lfp}^{\subseteq}\, \overrightarrow{\mathrm{Post}}[\![C]\!]In, & (104)\\ \text{where} & \overrightarrow{\mathrm{Post}}[\![C]\!] &\in& \wp(\Sigma[\![P]\!]) \xrightarrow{\text{cjm}} \left(\wp(\Sigma[\![P]\!]) \xrightarrow{\text{cjm}} \wp(\Sigma[\![P]\!])\right), &\\ & \overrightarrow{\mathrm{Post}}[\![C]\!]In &\triangleq& \lambda X \centerdot In \cup \mathrm{post}[\tau[\![C]\!]]\, X\ . & \end{array}$$

It follows that we have to effectively compute a machine representable approximation to the least solution of a fixpoint equation.

## 13. Abstract Interpretation of Imperative Programs

The classical approach [13, 5], followed during the Marktoberdorf course, consists in expressing the reachable states in fixpoint form (104) and then in using fixpoint transfer theorems [13] to get, by static partitioning [5], a system of equations attaching precise (for program proving) or approximate (for automated program analysis) abstract assertions to labels or program points [13]. Any chaotic [10] or asynchronous [4] strategy can be used to solve the system of equations iteratively. In practice, the iteration strategy which consists in iteratively or recursively traversing the dependence graph of the system of equations in weak topological order [3] speeds up the convergence, often very significatively [32]. This approach is quite general in that it does not depend upon a specific programming language. However for the simplistic language considered in these notes, the iteration order naturally mimics the execution order, as expressed by the big step relation semantics of Fig. 13. This remark allows us to obtain the corresponding efficient recursive equation solver in a more direct and simpler purely computational way.

### 13.1 Fixpoint precise abstraction

The following fixpoint abstraction theorem [13] is used to derive a precise abstract semantics from a concrete one expressed in fixpoint form. Recall that the iteration order of $F$ is the least ordinal $\epsilon$ such that $F^{\epsilon}(\bot) = \mathrm{lfp}^{\leq} F$, if it exists.

**Theorem 2** *If $\langle M,\ \preceq,\ 0,\ \vee\rangle$ is a cpo, the pair $\langle\alpha,\ \gamma\rangle$ is a Galois connection $\langle M,\ \preceq\rangle \xleftarrow[\alpha]{\gamma}\!\!\!\rightarrow \langle L,\ \sqsubseteq\rangle$, $\mathcal{F} \in M \stackrel{\text{mon}}{\longmapsto} M$ and $\mathcal{G} \in L \stackrel{\text{mon}}{\longmapsto} L$ are monotonic and*

$$\forall x \in M : x \preceq \mathit{lfp}^{\preceq}\, \mathcal{F} \implies \alpha \circ \mathcal{F}(x) = \mathcal{G} \circ \alpha(x)$$

*then*

$$\alpha(\mathit{lfp}^{\preceq}\, \mathcal{F}) \;=\; \mathit{lfp}^{\sqsubseteq}\, \mathcal{G}$$

*and the iteration order of $\mathcal{G}$ is less than or equal to that of $\mathcal{F}$ .*

*Proof* Since 0 is the infimum of $M$ and $\mathcal{F}$ is monotone, the transfinite iteration sequence $\mathcal{F}^{\delta}(0)$, $\delta \in \mathbb{O}$ (1) starting from 0 for $\mathcal{F}$ is an increasing chain which is strictly increasing and ultimately stationary and converges to $\mathcal{F}^{\epsilon} = \mathrm{lfp}^{\leq} F$ where $\epsilon$ is the iteration order of $\mathcal{F}$ (see [12]). It follows that $\forall\delta \in \mathbb{O} : \mathcal{F}^{\delta}(0) \leq \mathrm{lfp}^{\leq} F$ so

$$\alpha \circ \mathcal{F}(\mathcal{F}^{\delta}(0)) \;=\; \mathcal{G} \circ \alpha(\mathcal{G}^{\delta}(\bot))\ . \tag{105}$$

0 is the infimum of $M$ so $\forall y \in L : 0 \leq \gamma(y)$ whence $\alpha(0) \sqsubseteq y$ for Galois connections (8) proving that $\bot = \alpha(0)$ is the infimum of $L$. Let $\mathcal{G}^{\delta}(\bot)$, $\delta \in \mathbb{O}$ be the transfinite iteration sequence (1) starting from $\bot$ for $\mathcal{G}$. It is increasing and convergent to $\mathcal{G}^{\varepsilon} = \mathrm{lfp}^{\sqsubseteq} \mathcal{G}$ where $\varepsilon$ is minimal (see [12]).

We have $\alpha(\mathcal{F}^{0}(0)) = \alpha(0) = \bot = \mathcal{G}^{0}(\bot)$. If by induction hypothesis $\mathcal{F}^{\delta}(0) = \gamma(\mathcal{G}^{\delta}(\bot))$ then

$$\begin{aligned}
& \alpha(\mathcal{F}^{\delta+1}(0)) \\
= \quad & \wr\text{def. (1) of the transfinite iteration sequence}\wr \\
& \alpha(\mathcal{F}(\mathcal{F}^{\delta}(0))) \\
= \quad & \wr\text{commutation property (105)}\wr \\
& \mathcal{G} \circ \alpha(\mathcal{G}^{\delta}(\bot)) \\
= \quad & \wr\text{def. (1) of the transfinite iteration sequence}\wr \\
& \mathcal{G}^{\delta+1}(\bot)\ .
\end{aligned}$$

If $\lambda$ is a limit ordinal and $\forall\delta < \lambda, \alpha(\mathcal{F}^{\delta}(0)) = \mathcal{G}^{\delta}(\bot)$ by induction hypothesis then by def. (1) of transfinite iteration sequences, def. of lubs and $\alpha$ preserves existing lubs in Galois connections, we have $\alpha(\mathcal{F}^{\lambda}(0)) = \alpha(\bigvee_{\delta<\lambda} \mathcal{F}^{\delta}(0)) = \bigsqcup_{\delta<\lambda} \alpha(\mathcal{F}^{\delta}(0)) = \bigsqcup_{\delta<\lambda} \mathcal{G}^{\delta}(\bot) = \mathcal{G}^{\lambda}(\bot)$. By transfinite induction $\forall\delta \in \mathbb{O} : \alpha(\mathcal{F}^{\delta}(0)) = \mathcal{G}^{\delta}(\bot)$ whence in particular $\alpha(\mathrm{lfp}^{\preceq} \mathcal{F}) = \alpha(\mathcal{F}^{\epsilon}(0)) = \alpha(\mathcal{F}^{\max(\epsilon,\varepsilon)}(0)) = \mathcal{G}^{\max(\epsilon,\varepsilon)}(\bot) = \mathcal{G}^{\varepsilon}(\bot) = \mathrm{lfp}^{\sqsubseteq} \mathcal{G}$.

We have $\mathcal{F}^{\epsilon}(0) = \mathrm{lfp}^{\preceq} \mathcal{F}$ so $\mathcal{G}^{\epsilon}(\bot) = \alpha(\mathcal{F}^{\epsilon}(0)) = = \alpha \circ \mathcal{F}(\mathcal{F}^{\epsilon}(0)) = \mathcal{G} \circ \alpha(\mathcal{F}^{\epsilon}(0)) = \mathcal{G}(\mathcal{G}^{\epsilon}(\bot))$ proving that a fixpoint of $\mathcal{G}$ is reached at rank $\epsilon$ so $\varepsilon \leq \epsilon$ i.e. the iteration order $\varepsilon$ of $\mathcal{G}$ is less than or equal to the one $\epsilon$ of $\mathcal{F}$. □

As a simple example of application, we have proved in Sec. 2. that $t^{\star} = \mathrm{lfp}^{\subseteq} \lambda r \bullet 1_S \cup r \circ t$ where $t \subseteq S \times S$ so that given $P \subseteq S$, we can apply Th. 2 with Galois connection (98) to get

$$
\begin{array}{ll}
& \lambda X \cdot \mathrm{post}[X]\, P \circ \lambda r \cdot 1_S \cup t \circ r \\
= & \quad \wr \text{def. function composition } \circ \wr \\
& \lambda X \cdot \mathrm{post}[1_S \cup t \circ X]\, P \\
= & \quad \wr \text{Galois connection (98) so that } \lambda X \cdot \mathrm{post}[X]\, P \text{ preserves joins} \wr \\
& \lambda X \cdot \mathrm{post}[1_S]\, P \cup \mathrm{post}[t \circ X]\, P \\
= & \quad \wr \text{(100) and (99)} \wr \\
& \lambda X \cdot P \cup \mathrm{post}[X](\mathrm{post}[t]\, P) \\
= & \quad \wr \text{def. function composition } \circ \wr \\
& \lambda X \cdot P \cup \mathrm{post}[t]\, X \circ \lambda X \cdot \mathrm{post}[X]\, P,
\end{array}
$$

thus proving (101) that is $\mathrm{post}[t^\star]\, P = \mathrm{lfp}^{\subseteq}\, \lambda X \cdot P \cup \mathrm{post}[t]\, X$.

### *13.2 Fixpoint approximation abstraction*

The following fixpoint abstraction theorem [13] is used to derive an approximate abstract semantics from a concrete one expressed in fixpoint form.

**Theorem 3** *If $\langle M, \preceq, 0, \vee\rangle$ is a cpo, the pair $\langle \alpha, \gamma\rangle$ is a Galois connection $\langle M, \preceq\rangle \xleftrightarrow[\alpha]{\gamma} \langle L, \sqsubseteq\rangle$, $\mathcal{F} \in M \xrightarrow{\mathrm{mon}} M$ and $\mathcal{G} \in L \xrightarrow{\mathrm{mon}} L$ are monotonic and*

$$
\begin{array}{ll}
& \forall y \in L : \gamma(y) \preceq \mathit{lfp}^{\preceq}\, \mathcal{F} \implies \alpha \circ \mathcal{F} \circ \gamma(y) \sqsubseteq \mathcal{G}(y) \\
\textit{or equivalently} & \forall x \in M : \gamma \circ \alpha(x) \preceq \mathit{lfp}^{\preceq}\, \mathcal{F} \implies \alpha \circ \mathcal{F}(x) \sqsubseteq \mathcal{G} \circ \alpha(x) \\
\textit{or equivalently} & \forall y \in L : \gamma(y) \preceq \mathit{lfp}^{\preceq}\, \mathcal{F} \implies \mathcal{F} \circ \gamma(y) \preceq \gamma \circ \mathcal{G}(y)
\end{array}
$$

*then*

$$
\begin{array}{llcl}
& \mathit{lfp}^{\preceq}\, \mathcal{F} & \preceq & \gamma(\mathit{lfp}^{\sqsubseteq}\, \mathcal{G}) \\
\textit{and equivalently} & \alpha(\mathit{lfp}^{\preceq}\, \mathcal{F}) & \sqsubseteq & \mathit{lfp}^{\sqsubseteq}\, \mathcal{G},
\end{array}
$$

*Proof* For the equivalence of the hypotheses, observe that (for all $x \in M$ and $y \in L$)

$$
\begin{array}{ll}
& \alpha \circ \mathcal{F} \circ \gamma(y) \sqsubseteq \mathcal{G}(y) \\
\implies & \quad \wr \text{def. (8) of Galois connections and def. of functional composition } \circ \wr \\
& \mathcal{F} \circ \gamma(y) \preceq \gamma \circ \mathcal{G}(y) \\
\implies & \quad \wr \text{letting } y = \alpha(x) \wr \\
& \mathcal{F} \circ \gamma \circ \alpha(x) \preceq \gamma \circ \mathcal{G} \circ \alpha(x) \\
\implies & \quad \wr \gamma \circ \alpha \text{ extensive, } \mathcal{F} \text{ monotone and } \preceq \text{ transitive} \wr \\
& \mathcal{F}(x) \preceq \gamma \circ \mathcal{G} \circ \alpha(x) \\
\implies & \quad \wr \text{def. (8) of Galois connections} \wr \\
& \alpha \circ \mathcal{F}(x) \sqsubseteq \mathcal{G} \circ \alpha(x) \\
\implies & \quad \wr \text{letting } x = \gamma(y) \wr \\
& \alpha \circ \mathcal{F} \circ \gamma(y) \sqsubseteq \mathcal{G} \circ \alpha \circ \gamma(y) \\
\implies & \quad \wr \alpha \circ \gamma \text{ reductive, } \mathcal{G} \text{ monotone, } \preceq \text{ transitive} \wr \\
& \alpha \circ \mathcal{F} \circ \gamma(y) \sqsubseteq \mathcal{G}(y)\ .
\end{array}
$$

Moreover if $\forall y \in L : \gamma(y) \preceq \mathrm{lfp}^{\preceq}\, \mathcal{F}$ then in particular for any $x \in M$ and $y = \alpha(x)$, we get $\gamma \circ \alpha(x) \preceq \mathrm{lfp}^{\preceq}\, \mathcal{F}$. Then $\forall y' \in L$, if $x = \gamma(y')$ we get $\gamma \circ \alpha \circ \gamma(y') \preceq \mathrm{lfp}^{\preceq}\, \mathcal{F}$ that is $\gamma(y') \preceq \mathrm{lfp}^{\preceq}\, \mathcal{F}$ since $\gamma \circ \alpha \circ \gamma = \gamma$ for Galois connections.

Let $\mathcal{F}^{\delta}(0)$, $\delta \in \mathbb{O}$ and $\mathcal{G}^{\delta}(\bot)$, $\delta \in \mathbb{O}$ be the respective transfinite iteration sequences (1) respectively starting from $0$ and $\alpha(0)$ for $\mathcal{F}$ and $\mathcal{G}$. Observe that for all $y \in L$, $0 \preceq \gamma(y)$ whence $\alpha(0) \sqsubseteq y$ proving that $\bot \triangleq \alpha(0)$ is the infimum of $L$. Consequently both transfinite iteration sequences are increasing and convergent, respectively at ranks $\epsilon$ and $\varepsilon$ to $\mathcal{F}^{\epsilon} = \operatorname{lfp}^{\preceq} \mathcal{F}$ and $\mathcal{G}^{\varepsilon} = \operatorname{lfp}^{\sqsubseteq} \mathcal{G}$ (see [12]).

We have $\mathcal{F}^{0}(0) = 0 \preceq \gamma(\bot) = \gamma(\mathcal{G}^{0}(\bot))$. If by induction hypothesis $\mathcal{F}^{\delta}(0) \preceq \gamma(\mathcal{G}^{\delta}(\bot))$ then

$$
\begin{array}{ll}
 & \mathcal{F}^{\delta+1}(0) \\
= & \quad \wr \text{def. (1) of the transfinite iteration sequence} \int \\
 & \mathcal{F}(\mathcal{F}^{\delta}(0)) \\
\preceq & \quad \wr [\text{induction hypothesis } \mathcal{F}^{\delta}(0) \preceq \gamma(\mathcal{G}^{\delta}(\bot)) \text{ and } \mathcal{F} \text{ monotone} \int \\
 & \mathcal{F}(\gamma(\mathcal{G}^{\delta}(\bot))) \\
\preceq & \quad \wr \text{all elements of the increasing chain } \mathcal{F}^{\delta}, \delta \in \mathbb{O} \text{ are } \preceq\text{-less than } \operatorname{lfp}^{\preceq} \mathcal{F} \text{ and hypothesis} \\
 & \quad\ \ \forall y \in L : \gamma(y) \preceq \operatorname{lfp}^{\preceq} \mathcal{F} \implies \mathcal{F} \circ \gamma(y) \preceq \gamma \circ \mathcal{G}(y) \int \\
 & \gamma(\mathcal{G}(\mathcal{G}^{\delta}(\bot))) \\
= & \quad \wr \text{def. (1) of the transfinite iteration sequence} \int \\
 & \gamma(\mathcal{G}^{\delta+1}(\bot)) .
\end{array}
$$

If $\lambda$ is a limit ordinal and $\forall \delta < \lambda$, $\mathcal{F}^{\delta}(0) \preceq \gamma(\mathcal{G}^{\delta}(\bot))$ by induction hypothesis then by def. (1) of transfinite iteration sequences, def. of lubs and $\gamma$ monotone, we have $\mathcal{F}^{\lambda}(0) = \bigvee_{\delta<\lambda} \mathcal{F}^{\delta}(0)$ $\preceq \bigvee_{\delta<\lambda} \gamma(\mathcal{G}^{\delta}(\bot)) \preceq \gamma\left(\bigvee_{\delta<\lambda} \mathcal{G}^{\delta}(\bot)\right) = \gamma(\mathcal{G}^{\lambda}(\bot))$. By transfinite induction $\forall \delta \in \mathbb{O} : \mathcal{F}^{\delta}(0) \preceq$ $\gamma(\mathcal{G}^{\delta}(\bot))$ whence in particular $\operatorname{lfp}^{\preceq} \mathcal{F} = \mathcal{F}^{\epsilon}(0) = \mathcal{F}^{\max(\epsilon,\varepsilon)}(0) \preceq \gamma(\mathcal{G}^{\max(\epsilon,\varepsilon)}(\bot)) = \gamma(\mathcal{G}^{\varepsilon}(\bot))$ $= \gamma(\operatorname{lfp}^{\sqsubseteq} \mathcal{G})$. By definition (8) of Galois connections, this is equivalent to $\alpha(\operatorname{lfp}^{\preceq} \mathcal{F}) \sqsubseteq \operatorname{lfp}^{\sqsubseteq} \mathcal{G}$. $\square$

### *13.3 Abstract invariants*

Abstract invariants approximate program invariants in the form of abstract environments attached to program points. The abstract environments assign an abstract value in some abstract domain $L$ (as specified in Sec. 5.) to each program variable whence specify an overapproximation of its possible runtime values when execution reaches that point. The abstract domain is therefore ($P \in \text{Prog}$):

$$
\begin{array}{rcll}
\text{AEnv}[\![P]\!] & \triangleq & \text{Var}[\![P]\!] \mapsto L , & \text{abstract environments;} \\
\text{ADom}[\![P]\!] & \triangleq & \text{in}_P[\![P]\!] \mapsto \text{AEnv}[\![P]\!], & \text{abstract invariants.}
\end{array}
$$

The correspondence with program invariants is specified by the Galois connection

$$
\langle \wp(\Sigma[\![P]\!]), \subseteq \rangle \underset{\ddot{\alpha}[\![P]\!]}{\overset{\ddot{\gamma}[\![P]\!]}{\leftrightarrows}} \langle \text{ADom}[\![P]\!], \ddot{\sqsubseteq} \rangle \tag{106}
$$

where (see def. (18) of $\dot{\alpha}$ and (19) of $\dot{\gamma}$ where $\mathbb{V}$ is $\text{Var}[\![P]\!]$):

$$
\ddot{\alpha}[\![P]\!] I \triangleq \lambda \ell \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \langle \ell, \rho \rangle \in I\}), \tag{107}
$$

$$
\ddot{\gamma}[\![P]\!] J \triangleq \{\langle \ell, \rho \rangle \mid \rho \in \dot{\gamma}(J_\ell)\}, \tag{108}
$$

$$
J \mathrel{\ddot{\sqsubseteq}} J' \triangleq \forall \ell \in \text{in}_P[\![P]\!] : \forall \mathsf{X} \in \text{Var}[\![P]\!] : J_\ell(\mathsf{X}) \sqsubseteq J'_\ell(\mathsf{X}) .
$$

It follows that $\langle \text{ADom}[\![P]\!], \ddot{\sqsubseteq}, \ddot{\bot}, \ddot{\top}, \ddot{\sqcup}, \ddot{\sqcap} \rangle$ is a complete lattice.

The generic implementation of nonrelational abstract invariants has the following signature

```
module type Abstract_Dom_Algebra_signature =
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  sig
    open Abstract_Syntax
    open Labels
    type aDom         (* complete lattice of abstract invariants *)
    type element = aDom
    val bot  : unit -> aDom            (* infimum               *)
    val join : aDom -> (aDom -> aDom) (* least upper bound     *)
    val leq  : aDom -> (aDom -> bool) (* approximation ordering *)
    (* substitution *)
    val get  : aDom -> label -> E(L).env          (* j(l)         *)
    val set  : aDom -> label -> E(L).env -> aDom (* j[l <- r]    *)
  end;;
```

### 13.4 *Abstract predicate transformers*

This abstraction is extended to predicate transformers thanks to the functional abstraction

$$\langle \wp(\Sigma[\![P]\!]) \overset{\text{cjm}}{\longmapsto} \wp(\Sigma[\![P]\!]),\ \dot{\subseteq}\rangle \underset{\dot{\ddot{\alpha}}[\![P]\!]}{\overset{\dot{\ddot{\gamma}}[\![P]\!]}{\leftrightarrows}} \langle \mathrm{ADom}[\![P]\!] \overset{\text{mon}}{\longmapsto} \mathrm{ADom}[\![P]\!],\ \dot{\ddot{\sqsubseteq}}\rangle \tag{109}$$

where

$$\begin{aligned}
\dot{\ddot{\alpha}}[\![P]\!]F &\triangleq \ddot{\alpha}[\![P]\!] \circ F \circ \ddot{\gamma}[\![P]\!], \qquad (110)\\
\dot{\ddot{\gamma}}[\![P]\!]G &\triangleq \ddot{\gamma}[\![P]\!] \circ G \circ \ddot{\alpha}[\![P]\!],\\
G \ \dot{\ddot{\sqsubseteq}}\ G' &\triangleq \forall J \in \mathrm{ADom}[\![P]\!] : \forall \ell \in \mathrm{in}_P[\![P]\!] : \forall \mathsf{X} \in \mathrm{Var}[\![P]\!] : G(J)_\ell(\mathsf{X}) \sqsubseteq G'(J)_\ell(\mathsf{X})\ .
\end{aligned}$$

### 13.5 *Generic forward nonrelational abstract interpretation of programs*

The calculational design of the generic nonrelational abstract reachable states semantics of programs ($P \in \mathrm{Prog}$)

$$\mathrm{APost}[\![P]\!] \quad \in \quad \mathrm{ADom}[\![P]\!] \overset{\text{mon}}{\longmapsto} \mathrm{ADom}[\![P]\!]$$

can now be defined as an overapproximation of the forward collecting semantics (28)

$$\dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![P]\!]) \quad \dot{\ddot{\sqsubseteq}} \quad \mathrm{APost}[\![P]\!]\ . \tag{111}$$

Starting from the formal specification $\dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![P]\!])$, we derive an effectively computable function $\mathrm{APost}[\![P]\!]$ satisfying (111) by calculus. This abstract semantics is generic that is parameterized by an abstract algebra representing the approximation of value properties and corresponding operations as defined in Sec. 8. and 10.. For conciseness of the notation, the parameterization by $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap\rangle$, $\langle\alpha,\ \gamma\rangle$, etc. will be left implicit, although when programming this must be made explicit. We proceed by structural induction on the components $\mathrm{Cmp}[\![P]\!]$ of $P$ as defined in Sec. 12.2, proving that for all $C \in \mathrm{Cmp}[\![P]\!]$

*monotony*

$$\mathrm{APost}[\![C]\!] \quad \in \quad \mathrm{ADom}[\![P]\!] \overset{\text{mon}}{\longmapsto} \mathrm{ADom}[\![P]\!],$$

*soundness*

$$\dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![C]\!]) \quad \dot{\ddot{\sqsubseteq}} \quad \mathrm{APost}[\![C]\!], \tag{112}$$

*locality*

$$\forall J \in \mathrm{ADom}[\![P]\!] : \forall \ell \in \mathrm{in}[\![P]\!] - \mathrm{in}_P[\![C]\!] : J_\ell \quad = \quad (\mathrm{APost}[\![C]\!]J)_\ell, \tag{113}$$

*dependence*

$$\forall J, J' \in \mathrm{ADom}[\![P]\!] : (\forall \ell \in \mathrm{in}_P[\![C]\!] : J_\ell = J'_\ell) \implies (\forall \ell \in \mathrm{in}_P[\![C]\!] : (\mathrm{APost}[\![C]\!]J)_\ell = (\mathrm{APost}[\![C]\!]J')_\ell)\ . \tag{114}$$

Intuitively the locality and dependence properties express that the postcondition of a command can only depend upon and affect the abstract local invariants attached to labels in the command.

1 — For programs $P = S\ \texttt{;;}$, this will ultimately allow us to conclude that

$$\begin{array}{ll}
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![P]\!]) \\
= & \quad \wr \text{def. (103) of } \mathrm{Post}[\![P]\!] \wr \\
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^\star[\![P]\!]]) \\
= & \quad \wr \text{program syntax of Sec. 12.1 so that } P = S\ \texttt{;;} \wr \\
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^\star[\![S\ \texttt{;;}]\!]]) \\
= & \quad \wr (96) \wr \\
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^\star[\![S]\!]]) \\
\dot{\ddot{\sqsubseteq}} & \quad \wr \text{induction hypothesis (112)} \wr \\
& \mathrm{APost}[\![S]\!] \\
= & \quad \wr \text{by letting } \mathrm{APost}[\![S\ \texttt{;;}]\!] \triangleq \mathrm{APost}[\![S]\!] \text{ and } P = S\ \texttt{;;} \wr \\
& \mathrm{APost}[\![P]\!]\ .
\end{array}$$

$\mathrm{APost}[\![P]\!] = \mathrm{APost}[\![S]\!]$ is obviously monotonic by induction hypothesis while (113) vacuously holds and (114) follows by equality. We go on by structural induction on $C \in \mathrm{Cmp}[\![P]\!]$, starting from the basic cases.

2 — Identity $C = \texttt{skip}$ where $\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$.

$$\begin{array}{ll}
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![\texttt{skip}]\!]) \\
= & \quad \wr \text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![\texttt{skip}]\!] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (103) of Post} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![\texttt{skip}]\!]] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{big step operational semantics (92)} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{skip}]\!]] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (98) so that post preserves joins} \wr \\
& \ddot{\alpha}[\![P]\!] \circ (\mathrm{post}[1_{\Sigma[\![P]\!]}] \,\dot{\cup}\, \mathrm{post}[\tau[\![\texttt{skip}]\!]]) \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins} \wr \\
& (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \,\dot{\ddot{\sqcup}}\, (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau[\![\texttt{skip}]\!]] \circ \ddot{\gamma}[\![P]\!]) \\
\dot{\ddot{\sqsubseteq}} & \quad \wr \text{(100) and (106) so that } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \text{ is reductive} \wr \\
& 1_{\mathrm{ADom}[\![P]\!]} \,\dot{\ddot{\sqcup}}\, (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau[\![\texttt{skip}]\!]] \circ \ddot{\gamma}[\![P]\!]) \\
= & \quad \wr \text{def. (107) of } \ddot{\alpha} \wr \\
& 1_{\mathrm{ADom}[\![P]\!]} \,\dot{\ddot{\sqcup}}\, \lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid \langle l,\ \rho\rangle \in \mathrm{post}[\tau[\![\texttt{skip}]\!]] \circ \ddot{\gamma}[\![P]\!](J)\}) \\
= & \quad \wr \text{def. (97) of post} \wr \\
& 1_{\mathrm{ADom}[\![P]\!]} \,\dot{\ddot{\sqcup}}\, \lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau[\![\texttt{skip}]\!]\}) \\
= & \quad \wr \text{def. (76) and (62) of } \tau[\![\texttt{skip}]\!] \wr \\
& 1_{\mathrm{ADom}[\![P]\!]} \,\dot{\ddot{\sqcup}}\, \lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid l = \ell' \wedge \langle \ell,\ \rho\rangle \in \ddot{\gamma}[\![P]\!](J)\}) \\
= & \quad \wr \text{def. (108) of } \ddot{\gamma} \wr
\end{array}$$

$\lambda J \centerdot J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \mathrel{?} \dot{\alpha}(\{\rho \mid \rho \in \dot{\gamma}(J_\ell)\}) ¿ \dot{\alpha}(\emptyset))$

$\ddot{\sqsubseteq}$ $\quad$ ≀Galois connection (17) so that $\dot{\alpha} \circ \dot{\gamma}$ is reductive≀

$\lambda J \centerdot J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \mathrel{?} J_\ell ¿ \dot{\alpha}(\emptyset))$

$=$ $\quad$ ≀Galois connection (17) so that $\dot{\alpha}(\emptyset)$ is the infimum≀

$\lambda J \centerdot \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \mathrel{?} J_{\ell'} \dot{\sqcup} J_\ell ¿ J_l)$

$=$ $\quad$ ≀def. substitution≀

$\lambda J \centerdot J[\ell' \leftarrow J_{\ell'} \dot{\sqcup} J_\ell]$

$\triangleq$ $\quad$ ≀by letting $\text{APost}[\![\texttt{skip}]\!] \triangleq \lambda J \centerdot J[\ell' \leftarrow J_{\ell'} \dot{\sqcup} J_\ell]$≀

$\text{APost}[\![\texttt{skip}]\!]$ .

Monotony and the locality (113) and dependence (114) properties are trivial.

3 — Assignment $C = \mathtt{X} := A$ where $\text{at}_P[\![C]\!] = \ell$ and $\text{after}_P[\![C]\!] = \ell'$.

$\dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![\mathtt{X} := A]\!])$

$=$ $\quad$ ≀def. (110) of $\dot{\ddot{\alpha}}[\![P]\!]$≀

$\ddot{\alpha}[\![P]\!] \circ \text{Post}[\![\mathtt{X} := A]\!] \circ \ddot{\gamma}[\![P]\!]$

$=$ $\quad$ ≀def. (103) of Post≀

$\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![\mathtt{X} := A]\!]] \circ \ddot{\gamma}[\![P]\!]$

$=$ $\quad$ ≀big step operational semantics (92)≀

$\ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]} \cup \tau[\![\mathtt{X} := A]\!]] \circ \ddot{\gamma}[\![P]\!]$

$=$ $\quad$ ≀Galois connection (98) so that post preserves joins≀

$\ddot{\alpha}[\![P]\!] \circ (\text{post}[1_{\Sigma[\![P]\!]}] \dot{\cup} \text{post}[\tau[\![\mathtt{X} := A]\!]]) \circ \ddot{\gamma}[\![P]\!]$

$=$ $\quad$ ≀Galois connection (106) so that $\ddot{\alpha}[\![P]\!]$ preserves joins≀

$(\ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \dot{\ddot{\sqcup}} (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau[\![\mathtt{X} := A]\!]] \circ \ddot{\gamma}[\![P]\!])$

$\ddot{\sqsubseteq}$ $\quad$ ≀(100) and (106) so that $\ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!]$ is reductive≀

$1_{\text{ADom}[\![P]\!]} \dot{\ddot{\sqcup}} (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau[\![\mathtt{X} := A]\!]] \circ \ddot{\gamma}[\![P]\!])$

$=$ $\quad$ ≀def. (107) of $\ddot{\alpha}$≀

$1_{\text{ADom}[\![P]\!]} \dot{\ddot{\sqcup}} \lambda J \centerdot \lambda l \in \text{in}_P[\![P]\!] \centerdot \dot{\alpha}(\{\rho \mid \langle l,\ \rho\rangle \in \text{post}[\tau[\![\mathtt{X} := A]\!]] \circ \ddot{\gamma}[\![P]\!](J)\})$

$=$ $\quad$ ≀def. (97) of post ≀

$1_{\text{ADom}[\![P]\!]} \dot{\ddot{\sqcup}} \lambda J \centerdot \lambda l \in \text{in}_P[\![P]\!] \centerdot \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau[\![\mathtt{X} := A]\!]\})$

$=$ $\quad$ ≀def. (76) and (63) of $\tau[\![\mathtt{X} := A]\!]$≀

$1_{\text{ADom}[\![P]\!]} \dot{\ddot{\sqcup}} \lambda J \centerdot \lambda l \in \text{in}_P[\![P]\!] \centerdot \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : l' = \ell \wedge l = \ell' \wedge \exists i \in \mathbb{I} : \rho = \rho'[\mathtt{X} \leftarrow i] \wedge \rho' \vdash A \Mapsto i\})$

$=$ $\quad$ ≀def. (108) of $\ddot{\gamma}[\![P]\!]$≀

$\lambda J \centerdot J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \mathrel{?} \dot{\alpha}(\{\rho[\mathtt{X} \leftarrow i] \mid \rho \in \dot{\gamma}(J_\ell) \wedge i \in \mathbb{I} \wedge \rho \vdash A \Mapsto i\}) ¿ \dot{\alpha}(\emptyset))$

$\ddot{\sqsubseteq}$ $\quad$ ≀Galois connection (17) so that $\dot{\alpha}$ is monotonic≀

$\lambda J \centerdot \text{let } V \supseteq \{i \mid \exists \rho \in \dot{\gamma}(J_\ell) : \rho \vdash A \Mapsto i\} \text{ in let } V' \supseteq V \cap \mathbb{I} \text{ in}$
$\quad J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \wedge V' \neq \emptyset \mathrel{?} \dot{\alpha}(\{\rho[\mathtt{X} \leftarrow i] \mid \rho \in \dot{\gamma}(J_\ell) \wedge i \in V'\}) ¿ \dot{\alpha}(\emptyset))$

$\ddot{\sqsubseteq}$ $\quad$ ≀$V' = \emptyset \Rightarrow V \cap \mathbb{I} = \emptyset \Rightarrow V \subseteq \mathbb{E}$ since $V \subseteq \mathbb{E} \cup \mathbb{I}$ and $\mathbb{E} \cap \mathbb{I} = \emptyset$ whence $V \not\subseteq \mathbb{E} \Rightarrow V' \neq \emptyset$ together with the monotony of $\dot{\alpha}$≀

$\lambda J \centerdot \text{let } V \supseteq \{i \mid \exists \rho \in \dot{\gamma}(J_\ell) : \rho \vdash A \Mapsto i\} \text{ in let } V' \supseteq V \cap \mathbb{I} \text{ in}$
$\quad J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \wedge V \not\subseteq \mathbb{E} \mathrel{?} \dot{\alpha}(\{\rho[\mathtt{X} \leftarrow i] \mid \rho \in \dot{\gamma}(J_\ell) \wedge i \in V'\}) ¿ \dot{\alpha}(\emptyset))$

$\ddot{\sqsubseteq}$ $\quad$ ≀Galois connection (9) so that $\gamma \circ \alpha$ is extensive, $\alpha$ is monotonic, $V = \gamma(v)$ and $V' = \gamma(v')$≀

$\lambda J \centerdot \text{let } v \sqsupseteq \alpha(\{i \mid \exists \rho \in \dot{\gamma}(J_\ell) : \rho \vdash A \Mapsto i\}) \text{ in let } v' \sqsupseteq \alpha(\gamma(v) \cap \mathbb{I}) \text{ in}$
$\quad J \ddot{\sqcup} \lambda l \in \text{in}_P[\![P]\!] \centerdot (l = \ell' \wedge \gamma(v) \not\subseteq \mathbb{E} \mathrel{?} \dot{\alpha}(\{\rho[\mathtt{X} \leftarrow i] \mid \rho \in \dot{\gamma}(J_\ell) \wedge i \in \gamma(v')\}) ¿ \dot{\alpha}(\emptyset))$

$\ddot{\sqsubseteq}$ $\quad$ ≀Galois connection (9) so that $\alpha$ is monotonic whence $\alpha(X \cap Y) \sqsubseteq \alpha(X) \sqcap \alpha(Y)$, $\alpha \circ \gamma$ is reductive and def. (18) of $\dot{\alpha}$≀

$$\lambda J\bullet \text{let } v \sqsupseteq \alpha(\{i \mid \exists \rho \in \dot{\gamma}(J_\ell) : \rho \vdash A \Mapsto i\}) \text{ in let } v' \sqsupseteq v \sqcap \alpha(\mathbb{I}) \text{ in}$$
$$J \mathbin{\ddot{\sqcup}} \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \gamma(v) \not\subseteq \mathbb{E} \mathrel{\textbf{?}} \lambda \mathsf{Y} \in \mathbb{V}\bullet \alpha(\{\rho[\mathsf{X} \leftarrow i](\mathsf{Y}) \mid \rho \in \dot{\gamma}(J_\ell) \wedge i \in \gamma(v')\}) \mathrel{\textbf{¿}} \dot{\alpha}(\emptyset))$$

$\ddot{\sqsubseteq}$ ≀def. $\rho[\mathsf{X} \leftarrow i]$, $\dot{\gamma}(J_\ell) = \emptyset$ implies $v = \bot$ whence $\gamma(v) \subseteq \mathbb{E}$, def. (36) of $?^{\triangleright}$ and $\mho(x) \triangleq \gamma(x) \subseteq \mathbb{E}$ʃ

$$\lambda J\bullet \text{let } v \sqsupseteq \alpha(\{i \mid \exists \rho \in \dot{\gamma}(J_\ell) : \rho \vdash A \Mapsto i\}) \text{ in let } v' \sqsupseteq v \sqcap ?^{\triangleright} \text{ in}$$
$$J \mathbin{\ddot{\sqcup}} \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \neg\mho(v) \mathrel{\textbf{?}} \lambda \mathsf{Y} \in \mathbb{V}\bullet(\mathsf{Y} = \mathsf{X} \mathrel{\textbf{?}} \alpha(\{i \mid i \in \gamma(v')\}) \mathrel{\textbf{¿}} \alpha(\{\rho(\mathsf{Y}) \mid \rho \in \dot{\gamma}(J_\ell)\})) \mathrel{\textbf{¿}} \alpha(\emptyset))$$

$\ddot{\sqsubseteq}$ ≀def. (28) of the forward collecting semantics Faexp of arithmetic expressions, Galois connection (9) so that $\alpha \circ \gamma$ is reductive, def. (19) of $\dot{\gamma}$ʃ

$$\lambda J\bullet \text{let } v \sqsupseteq \alpha \circ \text{Faexp} \circ \dot{\gamma}(J_\ell) \text{ in}$$
$$J \mathbin{\ddot{\sqcup}} \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \neg\mho(v) \mathrel{\textbf{?}} \lambda \mathsf{Y} \in \mathbb{V}\bullet(\mathsf{Y} = \mathsf{X} \mathrel{\textbf{?}} v \sqcap ?^{\triangleright} \mathrel{\textbf{¿}} \alpha(\{\rho(\mathsf{Y}) \mid \rho(\mathsf{Y}) \in \gamma(J_\ell(\mathsf{Y}))\})) \mathrel{\textbf{¿}} \alpha(\emptyset))$$

$\ddot{\sqsubseteq}$ ≀Galois connection (9) so that $\alpha \circ \gamma$ is reductive, $\bot = \alpha(\emptyset)$ʃ

$$\lambda J\bullet \text{let } v \sqsupseteq \alpha \circ \text{Faexp} \circ \dot{\gamma}(J_\ell) \text{ in}$$
$$J \mathbin{\ddot{\sqcup}} \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \neg\mho(v) \mathrel{\textbf{?}} \lambda \mathsf{Y} \in \mathbb{V}\bullet(\mathsf{Y} = \mathsf{X} \mathrel{\textbf{?}} v \sqcap ?^{\triangleright} \mathrel{\textbf{¿}} J_\ell(\mathsf{Y})) \mathrel{\textbf{¿}} \bot)$$

$\ddot{\sqsubseteq}$ ≀def. (30) of $\alpha^{\triangleright}$, def. of $J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}]$ʃ

$$\lambda J\bullet \text{let } v \sqsupseteq \alpha^{\triangleright}(\text{Faexp})(J_\ell) \sqcap ?^{\triangleright} \text{ in } J \mathbin{\ddot{\sqcup}} \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \neg\mho(v) \mathrel{\textbf{?}} J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}] \mathrel{\textbf{¿}} \bot)$$

$\ddot{\sqsubseteq}$ ≀soundness (33) of the abstract interpretation $\text{Faexp}^{\triangleright}[\![A]\!]$ of arithmetic expressions $A$ and def. of $\ddot{\sqcup}$ʃ

$$\lambda J\bullet \text{let } v = \text{Faexp}^{\triangleright}[\![A]\!](J_\ell) \text{ in } \lambda l \in \text{in}_P[\![P]\!]\bullet(l = \ell' \wedge \neg\mho(v) \mathrel{\textbf{?}} J_{\ell'} \mathbin{\dot{\sqcup}} J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}] \mathrel{\textbf{¿}} J_l)$$

$=$ ≀def. of $J[\ell' \leftarrow \rho]$ʃ

$$\lambda J\bullet \text{let } v = \text{Faexp}^{\triangleright}[\![A]\!](J_\ell) \text{ in } (\neg\mho(v) \mathrel{\textbf{?}} J[\ell' \leftarrow J_{\ell'} \mathbin{\dot{\sqcup}} J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}]] \mathrel{\textbf{¿}} J)$$

$\triangleq$ ≀by letting $\text{APost}[\![\mathsf{X} := A]\!] \triangleq \lambda J\bullet \text{let } v = \text{Faexp}^{\triangleright}[\![A]\!](J_\ell) \text{ in } (\mho(v) \mathrel{\textbf{?}} J \mathrel{\textbf{¿}} J[\ell' \leftarrow J_{\ell'} \mathbin{\dot{\sqcup}} J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}]])$ʃ

$$\text{APost}[\![\mathsf{X} := A]\!] \; .$$

Monotony is trivial by monotony of $\text{Faexp}^{\triangleright}[\![A]\!]$ and so are the locality (113) and dependence (114) properties by (57).

4 — Sequence $C_1 \; ; \; \ldots \; ; \; C_n$, $n > 0$.

$$\dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![C_1 \; ; \; \ldots \; ; \; C_n]\!])$$

$=$ ≀def. (110) of $\dot{\ddot{\alpha}}[\![P]\!]$ʃ

$$\ddot{\alpha}[\![P]\!] \circ \text{Post}[\![C_1 \; ; \; \ldots \; ; \; C_n]\!] \circ \ddot{\gamma}[\![P]\!]$$

$=$ ≀def. (103) of Postʃ

$$\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![C_1 \; ; \; \ldots \; ; \; C_n]\!]] \circ \ddot{\gamma}[\![P]\!]$$

$=$ ≀big step operational semantics (95)ʃ

$$\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![C_1]\!] \circ \ldots \circ \tau^\star[\![C_n]\!]] \circ \ddot{\gamma}[\![P]\!]$$

$=$ ≀distribution (99) of post over composition $\circ$ʃ

$$\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![C_n]\!]] \circ \ldots \circ \text{post}[\tau^\star[\![C_1]\!]] \circ \ddot{\gamma}[\![P]\!]$$

$\dot{\ddot{\sqsubseteq}}$ ≀monotony and Galois connection (109) so that $\ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!]$ is extensiveʃ

$$\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![C_n]\!]] \, \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \ldots \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \, \text{post}[\tau^\star[\![C_1]\!]] \circ \ddot{\gamma}[\![P]\!]$$

$=$ ≀def. (110) of $\dot{\ddot{\alpha}}[\![P]\!]$ʃ

$$\dot{\ddot{\alpha}}[\![P]\!](\text{post}[\tau^\star[\![C_n]\!]]) \circ \ldots \circ \dot{\ddot{\alpha}}[\![P]\!](\text{post}[\tau^\star[\![C_1]\!]])$$

$=$ ≀def. (103) of Postʃ

$$\dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![C_n]\!]) \circ \ldots \circ \dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![C_1]\!])$$

$\dot{\ddot{\sqsubseteq}}$ ≀monotony and induction hypothesis (112)ʃ

$$\text{APost}[\![C_n]\!] \circ \ldots \circ \text{APost}[\![C_1]\!]$$

$\triangleq$ ≀by letting $\text{APost}[\![C_1 \; ; \; \ldots \; ; \; C_n]\!] \triangleq \text{APost}[\![C_n]\!] \circ \ldots \circ \text{APost}[\![C_1]\!]$ʃ

$\quad \text{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]$ .

Monotony follows from the induction hypothesis and the definition of $\text{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]$ by composition of monotonic functions $\text{APost}[\![C_i]\!]$, $i = 1, \ldots, n$. The locality (113) and dependence (114) properties follow by induction hypothesis for the $\text{APost}[\![C_i]\!]$, $i = 1, \ldots, n$ whence for $\text{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]$ by definition (58) of $\text{in}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] = \bigcup_{i=1}^{n} \text{in}_P[\![C_i]\!]$.

5 — Conditional $C = \texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}$ where $\text{at}_P[\![C]\!] = \ell$ and $\text{after}_P[\![C]\!] = \ell'$.

5.1 — By (93), we will need an over approximation of

$$
\begin{array}{ll}
& \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (107) of } \ddot{\alpha}[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \langle l,\ \rho\rangle \in \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!](J)\}) \\
= & \quad \wr \text{def. (97) of post} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau^B\}) \\
= & \quad \wr \text{def. (93) of } \tau^B \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : l' = \ell \wedge l = \text{at}_P[\![S_t]\!] \wedge \rho = \rho' \wedge \rho' \vdash \\
& B \Rrightarrow \text{tt}\}) \\
= & \quad \wr \text{def. (108) of } \ddot{\gamma}[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ \dot{\alpha}(\{\rho \in \dot{\gamma}(J_\ell) \mid \rho \vdash B \Rrightarrow \text{tt}\}) \ \textbf{¿}\ \dot{\alpha}(\emptyset)) \\
= & \quad \wr \text{def. (50) of the collecting semantics Cbexp}[\![B]\!] \text{ of boolean expressions } B \text{ and } \dot{\perp} \triangleq \\
& \quad\ \dot{\alpha}(\emptyset) \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ \dot{\alpha} \circ \text{Cbexp}[\![B]\!] \circ \dot{\gamma}(J_\ell) \ \textbf{¿}\ \dot{\perp}) \\
= & \quad \wr \text{def. (52) of } \ddot{\alpha} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ \ddot{\alpha}(\text{Cbexp}[\![B]\!])(J_\ell) \ \textbf{¿}\ \dot{\perp}) \\
\dot{\sqsubseteq} & \quad \wr \text{soundness (53) of the abstract semantics Abexp}[\![B]\!] \text{ of boolean expressions} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ \text{Abexp}[\![B]\!](J_\ell) \ \textbf{¿}\ \dot{\perp}) \ . \qquad (115)
\end{array}
$$

It is now easy to derive an over approximation of

$$
\begin{array}{ll}
& \ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]} \cup \tau^B] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (98) so that post preserves joins} \wr \\
& \ddot{\alpha}[\![P]\!] \circ (\text{post}[1_{\Sigma[\![P]\!]}] \,\dot{\cup}\, \text{post}[\tau^B]) \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins} \wr \\
& (\ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \,\ddot{\sqcup}\, (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!]) \\
= & \quad \wr \text{by (100) post preserves identity} \wr \\
& (\ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!]) \,\ddot{\sqcup}\, (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!]) \\
\ddot{\sqsubseteq} & \quad \wr \text{by the Galois connection (106) so that is } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \text{ is reductive and previous lemma} \\
& \quad\ (115) \wr \\
& \lambda J \cdot J \,\dot{\sqcup}\, \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ \text{Abexp}[\![B]\!](J_\ell) \ \textbf{¿}\ \dot{\perp}) \\
= & \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \ \textbf{?}\ J_{\text{at}_P[\![S_t]\!]} \,\dot{\sqcup}\, \text{Abexp}[\![B]\!](J_\ell) \ \textbf{¿}\ J_l) \ . \qquad (116)
\end{array}
$$

5.2 — By (93), we will also need an over approximation of

$$
\begin{array}{ll}
& \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^t] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (107) of } \ddot{\alpha}[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \langle l,\ \rho\rangle \in \text{post}[\tau^t] \circ \ddot{\gamma}[\![P]\!](J)\}) \\
= & \quad \wr \text{def. (97) of post} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau^t\})
\end{array}
$$

$$
\begin{array}{ll}
= & \quad \wr \text{def. (93) of } \tau^t \wr \\
& \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\rho \mid \exists \langle l', \rho' \rangle \in \ddot{\gamma}[\![P]\!](J) : l' = \mathrm{after}_P[\![S_t]\!] \wedge \rho' = \rho \wedge l = \ell'\}) \\
= & \quad \wr \text{def. (108) of } \ddot{\gamma}[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \text{ ? } \dot{\alpha}(\{\rho \mid \rho \in \dot{\gamma}(J_{\mathrm{after}_P[\![S_t]\!]})\}) \text{ ¿ } \dot{\alpha}(\emptyset)) \\
= & \quad \wr \text{Galois connection (17) so that } \dot{\alpha} \circ \dot{\gamma} \text{ is reductive and } \dot{\perp} = \dot{\alpha}(\emptyset) \wr \\
& \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \text{ ? } J_{\mathrm{after}_P[\![S_t]\!]} \text{ ¿ } \dot{\perp}) \qquad (117)
\end{array}
$$

It is now easy to derive an over approximation of

$$
\begin{array}{ll}
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (98) so that post preserves joins} \wr \\
& \ddot{\alpha}[\![P]\!] \circ (\mathrm{post}[1_{\Sigma[\![P]\!]}] \dot{\cup} \mathrm{post}[\tau^t]) \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins} \wr \\
& (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \mathrel{\ddot{\sqcup}} (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^t] \circ \ddot{\gamma}[\![P]\!]) \\
= & \quad \wr \text{by (100) post preserves identity} \wr \\
& (\ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!]) \mathrel{\ddot{\sqcup}} (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^t] \circ \ddot{\gamma}[\![P]\!]) \\
\mathrel{\ddot{\sqsubseteq}} & \quad \wr \text{by the Galois connection (106) so that is } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \text{ is reductive and previous lemma (117)} \wr \\
& \lambda J \cdot J \mathrel{\dot{\sqcup}} \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \text{ ? } J_{\mathrm{after}_P[\![S_t]\!]} \text{ ¿ } \dot{\perp}) \\
= & \quad \wr \dot{\perp} \text{ is the infimum} \wr \\
& \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \text{ ? } J_{\ell'} \mathrel{\dot{\sqcup}} J_{\mathrm{after}_P[\![S_t]\!]} \text{ ¿ } J_l) \qquad (118)
\end{array}
$$

5.3 — By (93), for the **then** branch of the conditional, we will need an over approximation of

$$
\begin{array}{ll}
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{distribution (99) of post over } \circ \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \mathrm{post}[\tau^\star[\![S_t]\!]] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![S_t]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \\
& \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B] \circ \ddot{\gamma}[\![P]\!] \\
\mathrel{\ddot{\sqsubseteq}} & \quad \wr \text{lemma (116) and monotony} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![S_t]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \\
& \mathrm{at}_P[\![S_t]\!] \text{ ? } J_{\mathrm{at}_P[\![S_t]\!]} \mathrel{\dot{\sqcup}} \mathrm{Abexp}[\![B]\!](J_\ell) \text{ ¿ } J_l) \\
= & \quad \wr \text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \circ \dot{\ddot{\alpha}}[\![P]\!] \mathrm{post}[\tau^\star[\![S_t]\!]] \circ \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!] \text{ ? } \\
& J_{\mathrm{at}_P[\![S_t]\!]} \mathrel{\dot{\sqcup}} \mathrm{Abexp}[\![B]\!](J_\ell) \text{ ¿ } J_l) \\
= & \quad \wr \text{def (103) of Post} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \circ \dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![S_t]\!]) \circ \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!] \text{ ? } \\
& J_{\mathrm{at}_P[\![S_t]\!]} \mathrel{\dot{\sqcup}} \mathrm{Abexp}[\![B]\!](J_\ell) \text{ ¿ } J_l) \\
= & \quad \wr \text{induction hypothesis (112) and monotony} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^t] \circ \ddot{\gamma}[\![P]\!] \circ \mathrm{APost}[\![S_t]\!] \circ \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!] \text{ ? } \\
& J_{\mathrm{at}_P[\![S_t]\!]} \mathrel{\dot{\sqcup}} \mathrm{Abexp}[\![B]\!](J_\ell) \text{ ¿ } J_l) \\
\mathrel{\ddot{\sqsubseteq}} & \quad \wr \text{lemma (118)} \wr \\
& \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \text{ ? } J_{\ell'} \mathrel{\dot{\sqcup}} J_{\mathrm{after}_P[\![S_t]\!]} \text{ ¿ } J_l) \circ \mathrm{APost}[\![S_t]\!] \circ \lambda J \cdot \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \\
& \mathrm{at}_P[\![S_t]\!] \text{ ? } J_{\mathrm{at}_P[\![S_t]\!]} \mathrel{\dot{\sqcup}} \mathrm{Abexp}[\![B]\!](J_\ell) \text{ ¿ } J_l) \\
& \quad \wr \text{def. of the let construct} \wr
\end{array}
$$

$$\lambda J \bullet \text{let } J^{t'} = \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \text{at}_P[\![S_t]\!] \;?\; J_{\text{at}_P[\![S_t]\!]} \mathbin{\dot\sqcup} \text{Abexp}[\![B]\!](J_\ell) \;¿\; J_l) \text{ in} \\ \text{let } J^{t''} = \text{APost}[\![S_t]\!](J^{t'}) \text{ in} \\ \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \ell' \;?\; J^{t''}_{\ell'} \mathbin{\dot\sqcup} J^{t''}_{\text{after}_P[\![S_t]\!]} \;¿\; J^{t''}_l) \tag{119}$$

Observe that monotony follows by induction hypothesis and the locality (113) and dependence (114) properties by induction hypothesis and the labelling condition (59).

5.4 — Since the case of the **else** branch of the conditional is similar to (5.3), we can now come back to the calculational design of APost$[\![$**if** $B$ **then** $S_t$ **else** $S_f$ **fi**$]\!]$ as an upper approximation of

$\ddot{\dot\alpha}[\![P]\!](\text{Post}[\![\textbf{if } B \textbf{ then } S_t \textbf{ else } S_f \textbf{ fi}]\!])$

$=$ $\wr$def. (110) of $\ddot{\dot\alpha}[\![P]\!]\wr$

$\ddot\alpha[\![P]\!] \circ \text{Post}[\![\textbf{if } B \textbf{ then } S_t \textbf{ else } S_f \textbf{ fi}]\!] \circ \ddot\gamma[\![P]\!]$

$=$ $\wr$def. (103) of Post$\wr$

$\ddot\alpha[\![P]\!] \circ \text{post}[\tau^\star[\![\textbf{if } B \textbf{ then } S_t \textbf{ else } S_f \textbf{ fi}]\!]] \circ \ddot\gamma[\![P]\!]$

$=$ $\wr$big step operational semantics (93)$\wr$

$\ddot\alpha[\![P]\!] \circ \text{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup (1_{\Sigma[\![P]\!]} \cup \tau^{\bar B}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot\gamma[\![P]\!]$

$=$ $\wr$Galois connection (98) so that post preserves joins$\wr$

$\ddot\alpha[\![P]\!] \circ (\text{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \mathbin{\dot\cup} \text{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar B}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)]) \circ \ddot\gamma[\![P]\!]$

$=$ $\wr$Galois connection (106) so that $\ddot\alpha[\![P]\!]$ preserves joins$\wr$

$(\ddot\alpha[\![P]\!] \circ \text{post}[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \circ \ddot\gamma[\![P]\!]) \mathbin{\ddot{\dot\cup}} (\ddot\alpha[\![P]\!] \circ \text{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar B}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot\gamma[\![P]\!])$

$\mathrel{\ddot{\dot\sqsubseteq}}$ $\wr$lemma (5.3) and similar one for the **else** branch$\wr$

$$\lambda J \bullet \text{let } J^{t'} = \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \text{at}_P[\![S_t]\!] \;?\; J_{\text{at}_P[\![S_t]\!]} \mathbin{\dot\sqcup} \text{Abexp}[\![B]\!](J_\ell) \;¿\; J_l) \text{ in} \\ \text{let } J^{t''} = \text{APost}[\![S_t]\!](J^{t'}) \text{ in} \\ \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \ell' \;?\; J^{t''}_{\ell'} \mathbin{\dot\sqcup} J^{t''}_{\text{after}_P[\![S_t]\!]} \;¿\; J^{t''}_l) \\ \ddot{\dot\sqcup} \\ \text{let } J^{f'} = \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \text{at}_P[\![S_f]\!] \;?\; J_{\text{at}_P[\![S_f]\!]} \mathbin{\dot\sqcup} \text{Abexp}[\![T(\neg B)]\!](J_\ell) \;¿\; J_l) \text{ in} \\ \text{let } J^{f''} = \text{APost}[\![S_f]\!](J^{f'}) \text{ in} \\ \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \ell' \;?\; J^{f''}_{\ell'} \mathbin{\dot\sqcup} J^{f''}_{\text{after}_P[\![S_f]\!]} \;¿\; J^{f''}_l) \tag{120}$$

$=$ $\wr$by grouping similar terms$\wr$

$$\lambda J \bullet \text{let } J^{t'} = \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \text{at}_P[\![S_t]\!] \;?\; J_{\text{at}_P[\![S_t]\!]} \mathbin{\dot\sqcup} \text{Abexp}[\![B]\!](J_\ell) \;¿\; J_l) \\ \text{and } J^{f'} = \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \text{at}_P[\![S_f]\!] \;?\; J_{\text{at}_P[\![S_f]\!]} \mathbin{\dot\sqcup} \text{Abexp}[\![T(\neg B)]\!](J_\ell) \;¿\; J_l) \text{ in} \\ \text{let } J^{t''} = \text{APost}[\![S_t]\!](J^{t'}) \\ \text{and } J^{f''} = \text{APost}[\![S_f]\!](J^{f'}) \text{ in} \\ \lambda l \in \text{in}_P[\![P]\!] \bullet (l = \ell' \;?\; J^{t''}_{\ell'} \mathbin{\dot\sqcup} J^{t''}_{\text{after}_P[\![S_t]\!]} \mathbin{\dot\sqcup} J^{f''}_{\ell'} \mathbin{\dot\sqcup} J^{f''}_{\text{after}_P[\![S_f]\!]} \;¿\; J^{t''}_l \mathbin{\dot\sqcup} J^{f''}_l)$$

$=$ $\wr$by locality (113) and labelling scheme (59) so that in particular $J^{t''}_{\ell'} = J^{t'}_{\ell'} = J^t_{\ell'} = J^f_{\ell'} = J^{f'}_{\ell'} = J^{f''}_{\ell'}$ and APost$[\![S_t]\!]$ and APost$[\![S_f]\!]$ do not interfere$\wr$

$$\begin{aligned}
& \lambda J \cdot \text{let } J' = \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S_t]\!] \;?\; J_{\text{at}_P[\![S_t]\!]} \,\dot\sqcup\, \text{Abexp}[\![B]\!](J_\ell) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \mid l = \text{at}_P[\![S_f]\!] \;?\; J_{\text{at}_P[\![S_f]\!]} \,\dot\sqcup\, \text{Abexp}[\![T(\neg B)]\!](J_\ell) \\
& \qquad\qquad\qquad\qquad\qquad\qquad ¿\; J_l) \text{ in} \\
& \qquad \text{let } J'' = \text{APost}[\![S_t]\!] \circ \text{APost}[\![S_f]\!](J') \text{ in} \\
& \qquad\quad \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \ell' \;?\; J''_{\ell'} \,\dot\sqcup\, J''_{\text{after}_P[\![S_t]\!]} \,\dot\sqcup\, J''_{\text{after}_P[\![S_f]\!]} \;¿\; J''_l) \\
= \quad & \wr \text{by letting APost}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] \text{ be defined as in (129)} \wr \\
& \text{APost}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]
\end{aligned}$$

6 — Iteration $C = \texttt{while } B \texttt{ do } S \texttt{ od}$ where $\ell = \text{at}_P[\![C]\!]$ and $\ell' = \text{after}_P[\![C]\!]$.

6.1 — By (94), we will need an over approximation of

$$\begin{aligned}
& \dot{\ddot\alpha}[\![P]\!](\text{post}[\tau^B]) \\
= \quad & \wr \text{def. (110) of } \dot{\ddot\alpha}[\![P]\!] \wr \\
& \ddot\alpha[\![P]\!] \circ \text{post}[\tau^B] \circ \ddot\gamma[\![P]\!] \\
= \quad & \wr \text{def. (107) of } \ddot\alpha[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot\alpha(\{\rho \mid \langle l,\ \rho\rangle \in \text{post}[\tau^B] \circ \ddot\gamma[\![P]\!](J)\}) \\
= \quad & \wr \text{def. (97) of post} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot\alpha(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot\gamma[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau^B\}) \\
= \quad & \wr \text{def. (94) of } \tau^B \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot\alpha(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot\gamma[\![P]\!](J) : l' = \ell \wedge l = \text{at}_P[\![S]\!] \wedge \rho = \rho' \wedge \rho' \vdash B \Rrightarrow \\
& \text{tt}\}) \\
= \quad & \wr \text{def. (108) of } \ddot\gamma[\![P]\!] \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S]\!] \;?\; \dot\alpha(\{\rho \in \dot\gamma(J_\ell) \mid \rho \vdash B \Rrightarrow \text{tt}\}) \;¿\; \dot\alpha(\emptyset)) \\
= \quad & \wr \text{def. (50) of the collecting semantics Cbexp}[\![B]\!] \text{ of boolean expressions } B \text{ and } \dot\bot \triangleq \\
& \quad \dot\alpha(\emptyset) \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S]\!] \;?\; \dot\alpha \circ \text{Cbexp}[\![B]\!] \circ \dot\gamma(J_\ell) \;¿\; \dot\bot) \\
= \quad & \wr \text{def. (52) of } \ddot\alpha \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S]\!] \;?\; \ddot\alpha(\text{Cbexp}[\![B]\!])(J_\ell) \;¿\; \dot\bot) \\
\dot{\ddot\sqsubseteq} \quad & \wr \text{soundness (53) of the abstract semantics Abexp}[\![B]\!] \text{ of boolean expressions} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{at}_P[\![S]\!] \;?\; \text{Abexp}[\![B]\!](J_\ell) \;¿\; \dot\bot) \\
\triangleq \quad & \wr \text{by introducing the APost}^B[\![C]\!] \text{ notation where } C = \texttt{while } B \texttt{ do } S \texttt{ od} \wr \\
& \text{APost}^B[\![C]\!]\ .
\end{aligned} \tag{121}$$

6.2 — Similarly ($\ell = \text{at}_P[\![C]\!]$),

$$\begin{aligned}
& \dot{\ddot\alpha}[\![P]\!](\text{post}[\tau^{\bar B}]) \\
= \quad & \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot\alpha(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot\gamma[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau^{\bar B}\}) \\
= \quad & \wr \text{def. (94) of } \tau^{\bar B} \wr \\
& \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot \dot\alpha(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot\gamma[\![P]\!](J) : l' = \ell \wedge l = \text{after}_P[\![C]\!] \wedge \rho = \rho' \wedge \rho' \vdash \\
& T(\neg B) \Rrightarrow \text{tt}\}) \\
\dot{\ddot\sqsubseteq} \quad & \lambda J \cdot \lambda l \in \text{in}_P[\![P]\!] \cdot (l = \text{after}_P[\![C]\!] \;?\; \text{Abexp}[\![T(\neg B)]\!](J_\ell) \;¿\; \dot\bot) \\
\triangleq \quad & \wr \text{by introducing the APost}^{\bar B}[\![C]\!] \text{ notation where } C = \texttt{while } B \texttt{ do } S \texttt{ od} \wr \\
& \text{APost}^{\bar B}[\![C]\!]\ .
\end{aligned} \tag{122}$$

6.3 — By (94), we will also need an over approximation of ($\ell = \mathrm{at}_P[\![C]\!]$)

$$\dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^R])$$
$$= \quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr$$
$$\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!]$$
$$= \quad \wr\text{def. (107) of } \ddot{\alpha}[\![P]\!]\wr$$
$$\lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid \langle l,\ \rho\rangle \in \mathrm{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!](J)\})$$
$$= \quad \wr\text{def. (97) of post}\wr$$
$$\lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : \langle\langle l',\ \rho'\rangle,\ \langle l,\ \rho\rangle\rangle \in \tau^R\})$$
$$= \quad \wr\text{def. (94) of } \tau^R\wr$$
$$\lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet \dot{\alpha}(\{\rho \mid \exists \langle l',\ \rho'\rangle \in \ddot{\gamma}[\![P]\!](J) : l' = \mathrm{after}_P[\![S]\!] \wedge \rho' = \rho \wedge l = \ell\})$$
$$= \quad \wr\text{def. (108) of } \ddot{\gamma}[\![P]\!]\wr$$
$$\lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \ell \ \textbf{?}\ \dot{\alpha}(\{\rho \mid \rho \in \dot{\gamma}(J_{\mathrm{after}_P[\![S]\!]})\}) \ \textbf{¿}\ \dot{\alpha}(\emptyset))$$
$$= \quad \wr\text{Galois connection (17) so that } \dot{\alpha} \circ \dot{\gamma} \text{ is reductive and } \dot{\bot} = \dot{\alpha}(\emptyset)\wr$$
$$\lambda J \bullet \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \ell \ \textbf{?}\ J_{\mathrm{after}_P[\![S]\!]} \ \textbf{¿}\ \dot{\bot})$$
$$\triangleq \quad \wr\text{by introducing the } \mathrm{APost}^R[\![C]\!] \text{ notation where } C = \texttt{while } B \texttt{ do } S \texttt{ od}\wr$$
$$\mathrm{APost}^R[\![C]\!]\ . \tag{123}$$

6.4 — For the loop entry, we will need an over approximation of

$$\dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}])$$
$$= \quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr$$
$$\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}] \circ \ddot{\gamma}[\![P]\!]$$
$$= \quad \wr\text{Galois connection (98) so that post preserves joins}\wr$$
$$\ddot{\alpha}[\![P]\!] \circ (\mathrm{post}[1_{\Sigma[\![P]\!]}] \,\dot{\cup}\, \mathrm{post}[\tau^B \circ \tau^\star[\![S]\!]] \,\dot{\cup}\, \mathrm{post}[\tau^{\bar{B}}]) \circ \ddot{\gamma}[\![P]\!]$$
$$= \quad \wr\text{by (99) post distributes over } \circ\wr$$
$$\ddot{\alpha}[\![P]\!] \circ (\mathrm{post}[1_{\Sigma[\![P]\!]}] \,\dot{\cup} (\mathrm{post}[\tau^\star[\![S]\!]] \circ \mathrm{post}[\tau^B]) \,\dot{\cup}\, \mathrm{post}[\tau^{\bar{B}}]) \circ \ddot{\gamma}[\![P]\!]$$
$$= \quad \wr\text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins}\wr$$
$$(\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \ \ddot{\cup}\ (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![S]\!]] \circ \mathrm{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!]) \ \ddot{\cup}\ (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^{\bar{B}}] \circ \ddot{\gamma}[\![P]\!])$$
$$\dot{\ddot{\sqsubseteq}} \quad \wr\text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony, }\wr$$
$$(\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \ \ddot{\cup}\ (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![S]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!]) \ \ddot{\cup}\ (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^{\bar{B}}] \circ \ddot{\gamma}[\![P]\!])$$
$$\dot{\ddot{\sqsubseteq}} \quad \wr(100) \text{ so that } \mathrm{post}[1_{\Sigma[\![P]\!]}] \text{ is the identity, Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \text{ is reductive, def. (103) of } \mathrm{Post}[\![S]\!] \text{, def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr$$
$$1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\mathrm{mon}} \mathrm{ADom}[\![P]\!]} \ \dot{\ddot{\cup}}\ (\ddot{\alpha}[\![P]\!](\mathrm{Post}[\![S]\!]) \circ \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^B])) \ \dot{\ddot{\cup}}\ \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^{\bar{B}}])$$
$$\dot{\ddot{\sqsubseteq}} \quad \wr\text{lemma (121), lemma (122), induction hypothesis (112) and monotony}\wr$$
$$1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\mathrm{mon}} \mathrm{ADom}[\![P]\!]} \ \dot{\ddot{\cup}}\ (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]) \ \dot{\ddot{\cup}}\ \mathrm{APost}^{\bar{B}}[\![C]\!]\ . \tag{124}$$

6.5 — For the loop exit, we will need an over approximation of

$$\dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R])$$
$$= \quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr$$
$$\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R] \circ \ddot{\gamma}[\![P]\!]$$
$$= \quad \wr\text{Galois connection (98) so that post preserves joins}\wr$$
$$\ddot{\alpha}[\![P]\!] \circ (\mathrm{post}[1_{\Sigma[\![P]\!]}] \,\dot{\cup}\, \mathrm{post}[\tau^\star[\![S]\!] \circ \tau^R]) \circ \ddot{\gamma}[\![P]\!]$$

$$
\begin{aligned}
= &\quad \wr\text{by (99) post distributes over } \circ\wr \\
& \ddot{\alpha}[\![P]\!] \circ (\text{post}[1_{\Sigma[\![P]\!]}] \dot{\cup} (\text{post}[\tau^\star[\![S]\!]] \circ \text{post}[\tau^R])) \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins}\wr \\
& (\ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \ddot{\cup} (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![S]\!]] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!]) \\
\dot{\ddot{\sqsubseteq}} &\quad \wr\text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony}\wr \\
& (\ddot{\alpha}[\![P]\!] \circ \text{post}[1_{\Sigma[\![P]\!]}] \circ \ddot{\gamma}[\![P]\!]) \dot{\ddot{\cup}} (\ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![S]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!]) \\
\dot{\ddot{\sqsubseteq}} &\quad \wr\text{(100) so that post}[1_{\Sigma[\![P]\!]}] \text{ is the identity, Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \\
&\quad\ \text{is reductive, def. (103) of Post}[\![S]\!]\text{, def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr \\
& 1_{\text{ADom}[\![P]\!] \xrightarrow{\text{mon}} \text{ADom}[\![P]\!]} \dot{\ddot{\cup}} (\dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![S]\!]) \circ \dot{\ddot{\alpha}}[\![P]\!](\text{post}[\tau^R])) \\
\dot{\ddot{\sqsubseteq}} &\quad \wr\text{lemma (123), induction hypothesis (112) and monotony}\wr \\
& 1_{\text{ADom}[\![P]\!] \xrightarrow{\text{mon}} \text{ADom}[\![P]\!]} \dot{\ddot{\cup}} (\text{APost}[\![S]\!] \circ \text{APost}^R[\![C]\!]) \ .
\end{aligned}
\tag{125}
$$

Observe that in all cases (121), (122), (123), (124) and (125), monotony follows by induction hypothesis and the locality (113) and dependence (114) properties by induction hypothesis and the labelling condition (60).

6.6 — By (94), we will also need an over approximation of

$$
\begin{aligned}
& \dot{\ddot{\alpha}}[\![P]\!](\text{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star]) \\
= &\quad \wr\text{Church } \lambda\text{-notation}\wr \\
& \dot{\ddot{\alpha}}[\![P]\!](\lambda In \cdot \text{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star]\, In) \\
= &\quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr \\
& \ddot{\alpha}[\![P]\!] \circ \big(\lambda In \cdot \text{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star]\, In\big) \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{fixpoint characterization (101)}\wr \\
& \ddot{\alpha}[\![P]\!] \circ \Big(\lambda In \cdot \text{lfp}^{\subseteq}\, \lambda X \cdot In \cup \text{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R]\, X\Big) \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{def. application and composition } \circ\wr \\
& \lambda J \cdot \ddot{\alpha}[\![P]\!](\text{lfp}^{\subseteq}\, \lambda X \cdot \ddot{\gamma}[\![P]\!](J) \cup \text{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R]\, X)
\end{aligned}
$$

In order to apply Th. 3, we compute

$$
\begin{aligned}
& \ddot{\alpha}[\![P]\!] \circ \lambda X \cdot \ddot{\gamma}[\![P]\!](J) \cup \text{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R]\, X \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{Church } \lambda\text{-notation}\wr \\
& \lambda X \cdot \ddot{\alpha}[\![P]\!](\ddot{\gamma}[\![P]\!](J) \cup \text{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R](\ddot{\gamma}[\![P]\!](X))) \\
= &\quad \wr\text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins}\wr \\
& \lambda X \cdot \ddot{\alpha}[\![P]\!](\ddot{\gamma}[\![P]\!](J)) \ddot{\cup}\, \ddot{\alpha}[\![P]\!](\text{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R](\ddot{\gamma}[\![P]\!](X))) \\
= &\quad \wr\text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \text{ is reductive and by (99) post distributes} \\
&\quad\ \text{over } \circ\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \text{post}[\tau^\star[\![S]\!]] \circ \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony}\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![S]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^B] \circ \ddot{\gamma}[\![P]\!] \\
= &\quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![S]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \dot{\ddot{\alpha}}[\![P]\!](\text{post}[\tau^B]) \\
= &\quad \wr\text{lemma (121) and monotony}\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^\star[\![S]\!]] \circ \ddot{\gamma}[\![P]\!] \circ \text{APost}^B[\![C]\!] \\
= &\quad \wr\text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \text{ and def. (103) of Post}\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!] \circ \dot{\ddot{\alpha}}[\![P]\!](\text{Post}[\![S]\!]) \circ \text{APost}^B[\![C]\!] \\
\dot{\ddot{\sqsubseteq}} &\quad \wr\text{induction hypothesis (112) and monotony, def. (110) of } \dot{\ddot{\alpha}}[\![P]\!]\wr \\
& \lambda X \cdot J \dot{\ddot{\cup}}\, \ddot{\alpha}[\![P]\!] \circ \text{post}[\tau^R] \circ \ddot{\gamma}[\![P]\!] \circ \text{APost}[\![S]\!] \circ \text{APost}^B[\![C]\!]
\end{aligned}
$$

$$
\begin{aligned}
= & \quad \wr \text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \wr \\
& \lambda X \bullet\, J \mathbin{\dot{\ddot{\sqcup}}} \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[\tau^R]) \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!] \\
= & \quad \wr \text{lemma (123)} \wr \\
& \lambda X \bullet\, J \mathbin{\dot{\ddot{\sqcup}}} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]
\end{aligned}
$$

so that we conclude

$$
\begin{aligned}
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star]) \\
= & \ \lambda J \bullet \ddot{\alpha}[\![P]\!](\mathrm{lfp}^{\subseteq}\, \lambda X \bullet \ddot{\gamma}[\![P]\!](J) \cup \mathrm{post}[\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R]\, X) \\
= & \quad \wr \text{Th. 3} \wr \\
= & \ \lambda J \bullet \mathrm{lfp}^{\ddot{\sqsubseteq}}\, \lambda X \bullet\, J \mathbin{\ddot{\sqcup}} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](X)
\end{aligned} \tag{126}
$$

Monotony follows when taking the least fixpoint of a functional which by induction hypothesis, is monotonic. The locality (113) and dependence (114) properties can be proved by induction hypothesis and the labelling condition (60) for all fixpoint iterates and is preserved by lubs whence when passing to the limit.

6.7 — We can now come back to the calculational design of APost[[**while** $B$ **do** $S$ **od**]] as an upper approximation of

$$
\begin{aligned}
& \dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![\mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}]\!]) \\
= & \quad \wr \text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![\mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}]\!] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (103) of Post} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![\mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}]\!]] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{big step operational semantics (94) of the iteration} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}})] \\
& \quad \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{distribution (99) of post over } \circ \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}] \circ \mathrm{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star] \circ \\
& \quad \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R] \circ \ddot{\gamma}[\![P]\!] \\
\dot{\ddot{\sqsubseteq}} & \quad \wr \text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony} \wr \\
& \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star] \circ \ddot{\gamma}[\![P]\!] \circ \\
& \quad \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R] \circ \ddot{\gamma}[\![P]\!] \\
\dot{\ddot{\sqsubseteq}} & \quad \wr \text{lemmata (124), (126), (125) and monotony} \wr \\
& (1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\text{mon}} \mathrm{ADom}[\![P]\!]} \mathbin{\dot{\ddot{\sqcup}}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]) \mathbin{\dot{\ddot{\sqcup}}} \mathrm{APost}^{\bar{B}}[\![C]\!]) \circ \lambda J \bullet \mathrm{lfp}^{\ddot{\sqsubseteq}}\, \lambda X \bullet\, J \mathbin{\ddot{\sqcup}} \\
& \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](X) \circ (1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\text{mon}} \mathrm{ADom}[\![P]\!]} \mathbin{\dot{\ddot{\sqcup}}} (\mathrm{APost}[\![S]\!] \circ \\
& \mathrm{APost}^R[\![C]\!])) \\
\triangleq & \ \mathrm{APost}[\![\mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}]\!] \,.
\end{aligned}
$$

In conclusion the calculational design of the generic forward nonrelational abstract semantics of programs leads to the functional and compositional characterization given in Fig. 14. In order to effectively compute an overapproximation of the set $\mathrm{post}[\tau^\star[\![P]\!]]\, \mathit{In}$ of states which are reachable by repeated small steps of the program $P$ from some given set $\mathit{In}$ of initial states, we use an overapproximation of the initial states

$$
\ddot{\alpha}[\![P]\!](\mathit{In}) \quad \ddot{\sqsubseteq} \quad I \tag{133}
$$

- $\mathrm{APost}[\![\mathtt{skip}]\!] \;=\; \lambda J\boldsymbol{\cdot} J[\mathrm{after}_P[\![\mathtt{skip}]\!] \leftarrow J_{\mathrm{after}_P[\![\mathtt{skip}]\!]} \dot{\sqcup} J_{\mathrm{at}_P[\![\mathtt{skip}]\!]}]$ (127)

- $\mathrm{APost}[\![\mathsf{X} := A]\!] \;=\; \lambda J\boldsymbol{\cdot}\mathrm{let}\ \ell = \mathrm{at}_P[\![\mathsf{X} := A]\!],\ \ell' = \mathrm{after}_P[\![\mathsf{X} := A]\!]\ \mathrm{in}$ (128)
$$\mathrm{let}\ v = \mathrm{Faexp}^{\triangleright}[\![A]\!](J_\ell)\ \mathrm{in}$$
$$(\mho(v)\ \mathbf{?}\ J\ \mathbf{¿}\ J[\ell' \leftarrow J_{\ell'} \dot{\sqcup} J_\ell[\mathsf{X} \leftarrow v \sqcap ?^{\triangleright}]])$$
where:
$$\forall v \in L : \mho(v) \implies \gamma(v) \subseteq \mathbb{E}$$

- $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi},\quad \mathrm{APost}[\![C]\!] \;=$ (129)
$$\lambda J\boldsymbol{\cdot}\mathrm{let}\ J' = J[\mathrm{at}_P[\![S_t]\!] \leftarrow J_{\mathrm{at}_P[\![S_t]\!]} \dot{\sqcup} \mathrm{Abexp}[\![B]\!](J_{\mathrm{at}_P[\![C]\!]});$$
$$\mathrm{at}_P[\![S_f]\!] \leftarrow J_{\mathrm{at}_P[\![S_f]\!]} \dot{\sqcup} \mathrm{Abexp}[\![T(\neg B)]\!](J_{\mathrm{at}_P[\![C]\!]})]\ \mathrm{in}$$
$$\mathrm{let}\ J'' = \mathrm{APost}[\![S_t]\!] \circ \mathrm{APost}[\![S_f]\!](J')\ \mathrm{in}$$
$$J''[\mathrm{after}_P[\![C]\!] \leftarrow J''_{\mathrm{after}_P[\![C]\!]} \dot{\sqcup} J''_{\mathrm{after}_P[\![S_t]\!]} \dot{\sqcup} J''_{\mathrm{after}_P[\![S_f]\!]}]$$

- $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od},\quad \mathrm{APost}[\![C]\!] \;=$ (130)
$$\left(1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\mathrm{mon}} \mathrm{ADom}[\![P]\!]} \ddot{\dot{\sqcup}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]) \ddot{\dot{\sqcup}} \mathrm{APost}^{\bar{B}}[\![C]\!]\right) \circ$$
$$\left(\lambda J\boldsymbol{\cdot}\mathrm{lfp}^{\ddot{\sqsubseteq}}\ \lambda X\boldsymbol{\cdot} J \ddot{\sqcup} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](X)\right) \circ$$
$$\left(1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\mathrm{mon}} \mathrm{ADom}[\![P]\!]} \ddot{\dot{\sqcup}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^R[\![C]\!])\right)$$
where:
$$\mathrm{APost}^B[\![C]\!] \;\stackrel{\Delta}{=}\; \lambda J\boldsymbol{\cdot}\ddot{\bot}[\mathrm{at}_P[\![S]\!] \leftarrow \mathrm{Abexp}[\![B]\!] J_{\mathrm{at}_P[\![C]\!]}]$$
$$\mathrm{APost}^{\bar{B}}[\![C]\!] \;\stackrel{\Delta}{=}\; \lambda J\boldsymbol{\cdot}\ddot{\bot}[\mathrm{after}_P[\![C]\!] \leftarrow \mathrm{Abexp}[\![T(\neg B)]\!] J_{\mathrm{at}_P[\![C]\!]}]$$
$$\mathrm{APost}^R[\![C]\!] \;\stackrel{\Delta}{=}\; \lambda J\boldsymbol{\cdot}\ddot{\bot}[\mathrm{at}_P[\![C]\!] \leftarrow J_{\mathrm{after}_P[\![S]\!]}]$$

- $\mathrm{APost}[\![C_1\ ;\ \ldots\ ;\ C_n]\!] \;=\; \mathrm{APost}[\![C_n]\!] \circ \ldots \circ \mathrm{APost}[\![C_1]\!]$ (131)

- $\mathrm{APost}[\![S\ ;;]\!] \;=\; \mathrm{APost}[\![S]\!]\ .$ (132)

Figure 14: Generic forward nonrelational reachability abstract semantics of programs

and compute $\mathrm{APost}[\![P]\!]I$. Elements of $\mathrm{ADom}[\![P]\!]$ must therefore be machine representable, which is obviously the case if the lattice $L$ of abstract value properties is itself machine representable. Moreover the computation of $\mathrm{APost}[\![P]\!]I$ terminates if the complete lattice $\langle L,\ \sqsubseteq\rangle$ satisfies the ascending chain condition. Otherwise convergence must be accelerated using widening/narrowing techniques [9]. The soundness of the approach is easily established

$$\begin{aligned}
&\mathrm{APost}[\![P]\!]I\\
\ddot{\sqsupseteq}\quad &\wr\text{soundness (111)}\wr\\
&\ddot{\alpha}[\![P]\!](\mathrm{Post}[\![P]\!])I\\
\ddot{\sqsupseteq}\quad &\wr\text{abstraction (133) of the entry condition and monotony}\wr\\
&\ddot{\alpha}[\![P]\!](\mathrm{Post}[\![P]\!])\,\ddot{\alpha}[\![P]\!](In)\\
\ddot{\sqsupseteq}\quad &\wr\text{def. (110)}\wr\\
&\ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![P]\!] \circ \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!](In)\\
\ddot{\sqsupseteq}\quad &\wr\text{Galois connection (106) so that } \ddot{\gamma}[\![P]\!] \circ \ddot{\alpha}[\![P]\!] \text{ is extensive and monotony}\wr\\
&\ddot{\alpha}[\![P]\!](\mathrm{Post}[\![P]\!](In))
\end{aligned}$$

[9] which were explained in the course but not, for short, in the notes, see [9, 16].

or equivalently, by the Galois connection (106)

$$\text{post}[\tau^\star[\![P]\!]]\, In \;\subseteq\; \ddot{\gamma}[\![P]\!](\text{APost}[\![P]\!]I)\ . \tag{134}$$

Notice that the set $\ddot{\gamma}[\![P]\!](\text{APost}[\![P]\!]I)$ is usually infinite so that its exploitation must be programmed using the encoding used for $\text{ADom}[\![P]\!]$ (or some machine representable image).

### *13.6 The generic abstract interpreter for reachability analysis*

The abstract syntax of commands is as follows

```
type com =
  | SKIP of label * label
  | ASSIGN of label * variable * aexp * label
  | SEQ of label * (com list) * label
  | IF of label * bexp * bexp * com * com * label
  | WHILE of label * bexp * bexp * com * label
```

For a command $C$, the first label $\text{at}_P[\![C]\!]$ (written `(at C)`) and the second $\text{after}_P[\![C]\!]$ (written `(after C)`) satisfy the labelling conditions of Sec. 12.3. The boolean expression $B$ of conditional and iteration commands is recorded by $T(B)$ and $T(\neg(B))$ as defined in Sec. 9.1.

The signature of the generic abstract interpreter [7] is

```
module type APost_signature =
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  functor (D: Abstract_Dom_Algebra_signature) ->
  functor (Faexp: Faexp_signature) ->
  functor (Baexp: Baexp_signature) ->
  functor (Abexp: Abexp_signature) ->
  sig
    open Abstract_Syntax
    (* generic forward nonrelational abstract reachability semantics of *)
 (* commands                                                          *)
    val aPost  : com -> D(L)(E).aDom -> D(L)(E).aDom
  end;;
```

Again the implementation is a prototype (in particular global operations on abstract invariants does not take the locality (113) and dependence properties (114) into account, a program optimization which is currently well beyond the current compiler technology for functional languages).

```
module APost_implementation =
 functor (L: Abstract_Lattice_Algebra_signature) ->
 functor (E: Abstract_Env_Algebra_signature) ->
 functor (D: Abstract_Dom_Algebra_signature) ->
 functor (Faexp: Faexp_signature) ->
 functor (Baexp: Baexp_signature) ->
 functor (Abexp: Abexp_signature) ->
 struct
  open Abstract_Syntax
  open Labels
  (* generic abstract environments                                     *)
  module E' = E(L)
  (* generic abstract invariants                                       *)
  module D' = D(L)(E)
  (* generic forward abstract interpretation of arithmetic operations  *)
```

```
  module Faexp' = Faexp(L)(E)
  (* generic [reductive] abstract interpretation of boolean operations *)
  module Abexp' = Abexp(L)(E)(Faexp)(Baexp)
  (* iterative fixpoint computation *)
  module F = Fixpoint((D':Poset_signature with type element=D(L)(E).aDom))
  (* generic forward nonrelational abstract reachability semantics     *)
  exception Error_aPost of string
  let rec aPost c j = match c with
  | (SKIP (l, l')) -> (D'.set j l' (E'.join (D'.get j l') (D'.get j l)))
  | (ASSIGN (l,x,a,l')) ->
     let v = (Faexp'.faexp a (D'.get j l))  in
      if (L.in_errors v) then j
      else (D'.set j l' (E'.join (D'.get j l') (E'.set (D'.get j l) x
                                              (L.meet v (L.f_RANDOM ())))))
  | (SEQ (l, s, l')) -> (aPostseq s j)
  | (IF (l, b, nb, t, f, l')) ->
     let j' = (D'.set j (at t) (E'.join (D'.get j (at t))
                                (Abexp'.abexp b  (D'.get j l)))) in
     let j'' = (D'.set j' (at f) (E'.join (D'.get j'
                       (at f)) (Abexp'.abexp nb (D'.get j' l)))) in
     let j''' = (aPost t (aPost f j'')) in
       (D'.set j''' l' (E'.join (E'.join (D'.get j''' l')
     (D'.get j''' (after t))) (D'.get j''' (after f))))
  | (WHILE (l, b, nb, c', l')) ->
     let aPostB j = (D'.set (D'.bot ()) (at c')
                            (Abexp'.abexp b (D'.get j l))) in
     let aPostnotB j = (D'.set (D'.bot ()) l'
                          (Abexp'.abexp nb (D'.get j l))) in
     let aPostR j = (D'.set (D'.bot ()) l (D'.get j (after c'))) in
     let j' = (D'.join j (aPost c' (aPostR j))) in
     let f x = (D'.join j' (aPostR (aPost c' (aPostB x)))) in
     let j'' = (F.lfp f (D'.bot ())) in
       (D'.join j'' (D'.join (aPost c' (aPostB j'')) (aPostnotB j'')))
  and aPostseq s j = match s with
  | []   -> raise (Error_aPost "empty sequence of commands")
  | [c]  -> (aPost c j)
  | h::t -> (aPostseq t (aPost h j))
 end;;

module APost = (APost_implementation:APost_signature);;
```

### *13.7 Abstract initial states*

We are left with the problem of defining the set *In* of initial states. More generally in the course we considered an assertion language allowing such safety and liveness non-trivial specifications. For short here, we consider the simple case when *In* is just the set $\mathrm{Entry}[\![P]\!]$ of program entry states (see (77))

$$
\begin{aligned}
& \ddot{\alpha}[\![P]\!](\mathrm{Entry}[\![P]\!]) \\
= \quad & \wr \text{def. (107) of } \ddot{\alpha}[\![P]\!] \text{ and (77) of } \mathrm{Entry}[\![P]\!] \,\wr \\
& \lambda \ell \in \mathrm{in}_P[\![P]\!] \cdot \dot{\alpha}(\{\lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \cdot \Omega_{\mathtt{i}} \mid \ell = \mathrm{at}_P[\![P]\!]\}) \\
= \quad & \wr \text{def. (18) of } \dot{\alpha} \,\wr \\
& \lambda \ell \in \mathrm{in}_P[\![P]\!] \cdot (\ell = \mathrm{at}_P[\![P]\!] \;\mathbf{?}\; \lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \cdot \alpha(\{\Omega_{\mathtt{i}}\}) \;\text{¿}\; \dot{\alpha}(\emptyset)) \\
= \quad & \wr \text{def. } \dot{\bot} \triangleq \dot{\alpha}(\emptyset),\ \ddot{\bot} \triangleq \lambda \ell \in \mathrm{in}_P[\![P]\!] \cdot \dot{\bot} \text{ and (16) of substitution} \wr \\
& \ddot{\bot}[\mathrm{at}_P[\![P]\!] \leftarrow \lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \cdot \alpha(\{\Omega_{\mathtt{i}}\})] \\
= \quad & \wr \text{by defining } \mathrm{AEntry}[\![P]\!] \triangleq \ddot{\bot}[\mathrm{at}_P[\![P]\!] \leftarrow \lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \cdot \alpha(\{\Omega_{\mathtt{i}}\})] \wr
\end{aligned} \tag{135}
$$

AEntry⟦$P$⟧ .

### 13.8 *Implementation of the abstract entry states*

The immediate translation is

```
module AEntry_implementation =
  functor (L: Abstract_Lattice_Algebra_signature) ->
  functor (E: Abstract_Env_Algebra_signature) ->
  functor (D: Abstract_Dom_Algebra_signature) ->
  struct
    open Abstract_Syntax
    open Labels
    (* generic abstract environments *)
    module E' = E(L)
    (* generic abstract invariants *)
    module D' = D(L)(E)
    (* abstraction of entry states *)
    exception Error_aEntry of string
    let aEntry c =
      if (at c) <> (entry ()) then
        raise (Error_aEntry "not the program entry point")
      else
        (D'.set (D'.bot ()) (at c) (E'.initerr ()))
  end;;
```

### 13.9 *The reachability static analyzer*

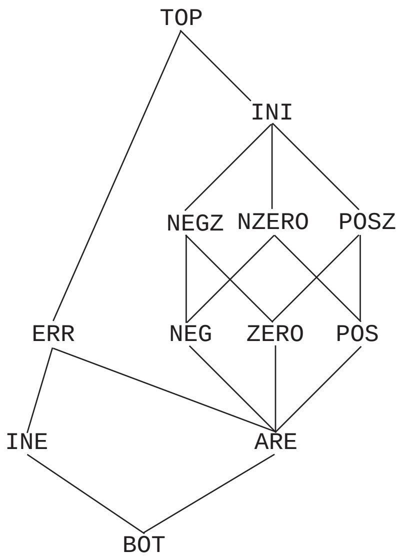
The generic abstract interpreter APost⟦$P$⟧(AEntry⟦$P$⟧) can be partially instantiated with (or without) reductive iterations, as follows [7]

```
module  Analysis_Reductive_Iteration_implementation =
 functor (L: Abstract_Lattice_Algebra_signature) ->
struct
 open Program_To_Abstract_Syntax
 module ENTRY = AEntry(L)(Abstract_Env_Algebra)(Abstract_Dom_Algebra)
 module POST = APost(L)(Abstract_Env_Algebra)(Abstract_Dom_Algebra)(Faexp)
                   (Baexp_Reductive_Iteration)(Abexp_Reductive_Iteration)
 module PRN = Pretty_Print(L)(Abstract_Env_Algebra)(Abstract_Dom_Algebra)
 let analysis () =
   print_string "type the  program to analyze..."; print_newline ();
   let p = abstract_syntax_of_program () in
     let j = (POST.aPost p (ENTRY.aEntry p)) in
    (PRN.pretty_print p j)
end;;
```

and then to a particular value property abstract domain

```
module ISS' = Analysis_Reductive_Iteration(ISS_Lattice_Algebra);;
```

Three examples of initialization and simple sign reachability analysis from the entry states are given below. The comparison of the first and second examples illustrates the loss of information due to the absence of an abstract value `POSZ` such that $\gamma(\texttt{POSZ}) \stackrel{\Delta}{=} [0, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\}$. The third example shows the imprecision on reachability resulting from the choice to have $\gamma()\texttt{BOT} \neq \emptyset$).




$$
\begin{array}{rcl}
\gamma(\texttt{BOT}) & \triangleq & \emptyset \\
\gamma(\texttt{INE}) & \triangleq & \{\Omega_{\texttt{i}}\} \\
\gamma(\texttt{ARE}) & \triangleq & \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{ERR}) & \triangleq & \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\} \\
\gamma(\texttt{NEG}) & \triangleq & [\texttt{min\_int}, -1] \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{ZERO}) & \triangleq & \{0, \Omega_{\texttt{a}}\} \\
\gamma(\texttt{POS}) & \triangleq & [1, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{NEGZ}) & \triangleq & [\texttt{min\_int}, 0] \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{NZERO}) & \triangleq & [\texttt{min\_int}, -1] \cup [1, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{POSZ}) & \triangleq & [0, \texttt{max\_int}] \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{INI}) & \triangleq & \mathbb{I} \cup \{\Omega_{\texttt{a}}\} \\
\gamma(\texttt{TOP}) & \triangleq & \mathbb{I}_{\Omega} \;=\; \mathbb{I} \cup \{\Omega_{\texttt{i}}, \Omega_{\texttt{a}}\}
\end{array}
$$

Figure 15: The lattice of errors and signs

```
{ n:ERR; i:ERR }
  n := ?; i := 1;
{ n:INI; i:POS }
  while (i < n) do
    { n:POS; i:POS }
      i := (i + 1)
    { n:POS; i:POS }
  od
{ n:INI; i:POS }
```

```
{ n:ERR; i:ERR }
  n := ?; i := 0;
{ n:INI; i:INI }
  while (i < n) do
    { n:INI; i:INI }
      i := (i + 1)
    { n:INI; i:INI }
  od
{ n:INI; i:INI }
```

```
{ x:ERR }
  x := (1 / 0);
{ x:BOT }
  skip;
{ x:BOT }
  x := 1
{ x:POS }
```

Precision can be increased to solve these problems by using the lattice of errors and signs specified in Fig. 15, as shown below.

```
{ n:ERR; i:ERR }
  n := ?; i := 0;
{ n:INI; i:POSZ }
  while (i < n) do
    { n:POS; i:POSZ }
      i := (i + 1)
    { n:POS; i:POS }
  od
{ n:INI; i:POSZ }
```

```
{ x:ERR }
  x := (1 / 0);
{ x:BOT }
  skip;
{ x:BOT }
  x := 1
{ x:BOT }
```

The next two examples (for which the gathered information is the same whether reductive iterations are used or not) show that the classical handling of arithmetic or boolean expressions using assignments of simple monomials to auxiliary variables (`i1` in the example below) is less precise than the algorithm proposed in these notes.

```
{ x:ERR; y:ERR }
  x := 0; y := ?;
{ x:ZERO; y:INI }
  while (x = -y) do
    { x:ZERO; y:ZERO }
      skip
    { x:ZERO; y:ZERO }
  od
{ x:ZERO; y:INI }
```

```
{ x:ERR; y:ERR; i1:ERR }
  x := 0; y := ?; i1 := -y;
{ x:ZERO; y:INI; i1:INI }
  while (x = i1) do
    { x:ZERO; y:INI; i1:ZERO }
      skip; i1 := -y
    { x:ZERO; y:INI; i1:INI }
  od
{ x:ZERO; y:INI; i1:INI }
```

The same loss of precision due to the nonrelational abstraction (17) appears when boolean expressions are analyzed by compilation into intermediate short-circuit conditional code

```
{ x:ERR; y:ERR; z:ERR }
  x := 0; y := ?; z := ?;
{ x:ZERO; y:INI; z:INI }
  if ((x=y)&((z+1)=x)&(y=z)) then


    { x:BOT; y:BOT; z:BOT }
      skip






  else
    { x:ZERO; y:INI; z:INI }
      skip
  fi
{ x:ZERO; y:INI; z:INI }
```

```
{ x:ERR; y:ERR; z:ERR }
  x := 0; y := ?; z := ?;
{ x:ZERO; y:INI; z:INI }
  if ((x=y)&((z+1)=x)) then
    { x:ZERO; y:ZERO; z:NEG }
      if (y=z) then
        { x:ZERO; y:BOT; z:BOT }
          skip
      else
        { x:ZERO; y:ZERO; z:NEG }
          skip
      fi
    { x:ZERO; y:ZERO; z:NEG }
  else
    { x:ZERO; y:INI; z:INI }
      skip
  fi
{ x:ZERO; y:INI; z:INI }
```

Similar examples can be provided for any nontrivial nonrelational abstract domain.

### *13.10 Specializing the abstract interpreter to reachability analysis from the entry states*

As a very first step towards efficient analyzers, the abstract interpreter of Fig. 14 can be specialized for reachability analysis from program entry states [7]. We want to calculate

$$\mathrm{APost}\llbracket P\rrbracket(\mathrm{AEntry}\llbracket P\rrbracket)$$

and more generally, for all program subcommands $C \in \mathrm{Cmp}\llbracket P\rrbracket$

$$\lambda r \in \mathrm{AEnv}\llbracket P\rrbracket \cdot \mathrm{APost}\llbracket C\rrbracket(\ddot{\bot}[\mathrm{at}_P\llbracket C\rrbracket \leftarrow r])$$

that is

$$\mathrm{APostEn}_P\llbracket C\rrbracket \quad\stackrel{\Delta}{=}\quad \alpha^{\varepsilon}_P\llbracket C\rrbracket(\mathrm{APost}\llbracket C\rrbracket) \tag{136}$$

where

$$\alpha^{\varepsilon}_P\llbracket C\rrbracket \quad\stackrel{\Delta}{=}\quad \lambda F\cdot\lambda r\cdot F(\ddot{\bot}[\mathrm{at}_P\llbracket C\rrbracket \leftarrow r]) \tag{137}$$

$$\gamma^{\varepsilon}_P\llbracket C\rrbracket \quad\stackrel{\Delta}{=}\quad \lambda f\cdot\lambda J\cdot(\forall l \neq \mathrm{at}_P\llbracket C\rrbracket : J_l = \dot{\bot}\ \textbf{?}\ f(J_{\mathrm{at}_P\llbracket C\rrbracket})\ \textbf{¿}\ \ddot{\top}) \tag{138}$$

is such that

$$\begin{array}{ll}
& \alpha^{\varepsilon}_P\llbracket C\rrbracket F \mathrel{\dot{\ddot{\sqsubseteq}}} f \\
\Longleftrightarrow & \wr\text{def. of the pointwise ordering } \dot{\ddot{\sqsubseteq}} \text{ and (137) of } \alpha^{\varepsilon}_P\llbracket C\rrbracket\wr \\
& \forall r \in \mathrm{AEnv}\llbracket P\rrbracket : F(\ddot{\bot}[\mathrm{at}_P\llbracket C\rrbracket \leftarrow r]) \mathrel{\ddot{\sqsubseteq}} f(r) \\
\Longleftrightarrow & \wr\text{for } \Rightarrow\text{, by choosing } r = J_{\mathrm{at}_P\llbracket C\rrbracket} \text{ and } \ddot{\top} \text{ is the supremum, while for } \Leftarrow\text{, by choosing} \\
& \quad J = \ddot{\bot}[\mathrm{at}_P\llbracket C\rrbracket \leftarrow r]\wr \\
& \forall J \in \mathrm{ADom}\llbracket P\rrbracket : F(J) \mathrel{\ddot{\sqsubseteq}} (\forall l \neq \mathrm{at}_P\llbracket C\rrbracket : J_l = \dot{\bot}\ \textbf{?}\ f(J_{\mathrm{at}_P\llbracket C\rrbracket})\ \textbf{¿}\ \ddot{\top}) \\
\Longleftrightarrow & \wr\text{def. (138) of } \gamma^{\varepsilon}_P\llbracket C\rrbracket\wr \\
& \forall J \in \mathrm{ADom}\llbracket P\rrbracket : F(J) \mathrel{\ddot{\sqsubseteq}} \gamma^{\varepsilon}_P\llbracket C\rrbracket(f)J \\
\Longleftrightarrow & \wr\text{def. of the pointwise ordering } \dot{\ddot{\sqsubseteq}} \text{ (which is overloaded)}\wr \\
& F \mathrel{\dot{\ddot{\sqsubseteq}}} \gamma^{\varepsilon}_P\llbracket C\rrbracket(f)
\end{array}$$

whence

$$\langle \mathrm{ADom}[\![P]\!] \stackrel{\mathrm{mon}}{\longmapsto} \mathrm{ADom}[\![P]\!],\ \dot{\ddot{\sqsubseteq}}\rangle \xrightleftharpoons[\alpha^{\varepsilon}_{P}[\![C]\!]]{\gamma^{\varepsilon}_{P}[\![C]\!]} \langle \mathrm{AEnv}[\![P]\!] \stackrel{\mathrm{mon}}{\longmapsto} \mathrm{ADom}[\![P]\!],\ \dot{\ddot{\sqsubseteq}}\rangle\ .$$

We consider the simple situation where $\gamma(\bot) = \emptyset$ for which (34), (39) and (54) do hold. It follows by structural and fixpoint induction that for all $C \in \mathrm{Cmp}[\![P]\!]$

$$\mathrm{APost}[\![C]\!](\ddot{\bot}) \quad = \quad \ddot{\bot}\ . \tag{139}$$

We calculate $\mathrm{APostEn}_P[\![C]\!]$ by structural induction and trivially prove simultaneously

$$\begin{array}{ll} \textit{locality} & \forall r \in \mathrm{AEnv}[\![P]\!] : \forall l \in \mathrm{in}[\![P]\!] - \mathrm{in}_P[\![C]\!] (\mathrm{APostEn}_P[\![C]\!]r)_l = \dot{\bot}\ , \quad (140)\\ \textit{extension} & \forall r \in \mathrm{AEnv}[\![P]\!] : r \sqsubseteq (\mathrm{APostEn}_P[\![C]\!]r)_{\mathrm{at}_P[\![C]\!]}\ . \quad (141)\end{array}$$

1 —— Identity $C = \mathtt{skip}$ where $\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$

$$\begin{array}{ll} & \mathrm{APostEn}_P[\![\mathtt{skip}]\!]\\ = & \quad \wr \text{def. (136) of } \mathrm{APostEn}_P \text{ and (137) of } \alpha^{\varepsilon}_{P} \wr\\ & \lambda r \cdot \mathrm{APost}[\![\mathtt{skip}]\!](\ddot{\bot}[\ell \leftarrow r])\\ = & \quad \wr \text{def. (127) of } \mathrm{APost}[\![\mathtt{skip}]\!]\text{, labelling condition (56) and substitution (16)} \wr\\ & \lambda r \cdot \ddot{\bot}[\ell \leftarrow r;\ \ell' \leftarrow r]\ . \end{array}$$

(140) and (141) hold because $\mathrm{in}_P[\![\mathtt{skip}]\!] = \{\ell, \ell'\}$ and reflexivity.

2 —— Assignment $C = \mathtt{X} := A$ where $\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$

$$\begin{array}{ll} & \mathrm{APostEn}_P[\![\mathtt{X} := A]\!]\\ = & \quad \wr \text{def. (136) of APostEn and (137) of } \alpha^{e} \wr\\ & \lambda r \cdot \mathrm{APost}[\![\mathtt{X} := A]\!](\ddot{\bot}[\ell \leftarrow r])\\ = & \quad \wr \text{def. (128) of } \mathrm{APost}[\![\mathtt{X} := A]\!] \wr\\ & \lambda r \cdot \text{let } v = \mathrm{Faexp}^{\triangleright}[\![A]\!]((\ddot{\bot}[\ell \leftarrow r])_\ell) \text{ in}\\ & \qquad (\mho(v)\ \mathbf{?}\ \ddot{\bot}[\ell \leftarrow r]\ \mathbf{¿}\ (\ddot{\bot}[\ell \leftarrow r])[\ell' \leftarrow (\ddot{\bot}[\ell \leftarrow r])_{\ell'} \dot{\sqcup} (\ddot{\bot}[\ell \leftarrow r])_\ell[\mathtt{X} \leftarrow v \sqcap ?^{\triangleright}]])\\ = & \quad \wr \text{labelling condition (56), } \dot{\bot} \text{ is the infimum and def. (16) of substitution} \wr\\ & \lambda r \cdot \text{let } v = \mathrm{Faexp}^{\triangleright}[\![A]\!](r) \text{ in } (\mho(v)\ \mathbf{?}\ \ddot{\bot}[\ell \leftarrow r]\ \mathbf{¿}\ \ddot{\bot}[\ell \leftarrow r;\ \ell' \leftarrow r[\mathtt{X} \leftarrow v \sqcap ?^{\triangleright}]])\ . \end{array}$$

(140) and (141) hold because $\mathrm{in}_P[\![\mathtt{X} := A]\!] = \{\ell, \ell'\}$ and reflexivity.

3 —— Conditional $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}$ where $\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$

$$\begin{array}{ll} & \mathrm{APostEn}_P[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!]\\ = & \quad \wr \text{def. (136) of APostEn and (137) of } \alpha^{e} \wr\\ & \lambda r \cdot \mathrm{APost}[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!](\ddot{\bot}[\ell \leftarrow r])\\ = & \quad \wr \text{def. (120) of } \mathrm{APost}[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!] \wr\\ & \lambda r \cdot \text{let } J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!]\ \mathbf{?}\ (\ddot{\bot}[\ell \leftarrow r])_{\mathrm{at}_P[\![S_t]\!]}\ \dot{\sqcup}\\ & \qquad\qquad \mathrm{Abexp}[\![B]\!]((\ddot{\bot}[\ell \leftarrow r])_\ell)\ \mathbf{¿}\ (\ddot{\bot}[\ell \leftarrow r])_l) \text{ in}\\ & \qquad \text{let } J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'}) \text{ in } \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell'\ \mathbf{?}\ J^{t''}_{\ell'} \dot{\sqcup} J^{t''}_{\mathrm{after}_P[\![S_t]\!]}\ \mathbf{¿}\ J^{t''}_l)\\ & \quad \ddot{\dot{\sqcup}}\\ & \quad \text{let } J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_f]\!]\ \mathbf{?}\ (\ddot{\bot}[\ell \leftarrow r])_{\mathrm{at}_P[\![S_f]\!]}\ \dot{\sqcup}\\ & \qquad\qquad \mathrm{Abexp}[\![T(\neg B)]\!]((\ddot{\bot}[\ell \leftarrow r])_\ell)\ \mathbf{¿}\ (\ddot{\bot}[\ell \leftarrow r])_l) \text{ in}\\ & \qquad \text{let } J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'}) \text{ in } \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell'\ \mathbf{?}\ J^{f''}_{\ell'} \dot{\sqcup} J^{f''}_{\mathrm{after}_P[\![S_f]\!]}\ \mathbf{¿}\ J^{f''}_l)\ . \end{array}$$

For the true alternative, by the labelling condition (59) so that $\ell \neq \mathrm{at}_P\llbracket S_t \rrbracket$ and def. (16) of substitution, we get

$$
\begin{array}{ll}
& \lambda r\bullet \mathrm{let}\ J^{t'} = \ddot{\bot}[\ell \leftarrow r;\ \mathrm{at}_P\llbracket S_t\rrbracket \leftarrow \mathrm{Abexp}\llbracket B\rrbracket r]\ \mathrm{in} \\
& \qquad \mathrm{let}\ J^{t''} = \mathrm{APost}\llbracket S_t\rrbracket(J^{t'})\ \mathrm{in}\ \lambda l \in \mathrm{in}_P\llbracket P\rrbracket\bullet(l = \ell'\ \mathbf{?}\ J^{t''}_{\ell'} \mathbin{\dot{\sqcup}} J^{t''}_{\mathrm{after}_P\llbracket S_t\rrbracket}\ \mathbf{¿}\ J^{t''}_{l}) \\
= & \quad \wr\text{by the locality (113) and dependence (114) properties, the labelling conditions (56)} \\
& \quad\ \ \text{so that } \ell \neq \ell' \text{ and (59) so that } \ell, \ell' \notin \mathrm{in}_P\llbracket S_t\rrbracket \wr \\
& \lambda r\bullet \mathrm{let}\ J^{t} = \mathrm{APost}\llbracket S_t\rrbracket(\ddot{\bot}[\mathrm{at}_P\llbracket S_t\rrbracket \leftarrow \mathrm{Abexp}\llbracket B\rrbracket r])\ \mathrm{in}\ \ddot{\bot}[\ell \leftarrow r;\ \ell' \leftarrow J^{t}_{\mathrm{after}_P\llbracket S_t\rrbracket}] \mathbin{\ddot{\sqcup}} J^{t} \\
= & \quad \wr\text{def. (136) of APostEn, (137) of } \alpha^e \text{ and induction hypothesis}\wr \\
& \lambda r\bullet \mathrm{let}\ J^{t} = \mathrm{APostEn}_P\llbracket S_t\rrbracket(\mathrm{Abexp}\llbracket B\rrbracket r)\ \mathrm{in}\ \ddot{\bot}[\ell \leftarrow r;\ \ell' \leftarrow J^{t}_{\mathrm{after}_P\llbracket S_t\rrbracket}] \mathbin{\ddot{\sqcup}} J^{t}\ ,
\end{array}
$$

so that we get (147) by grouping with a similar result for the false alternative and using the labelling condition (59). (140) and (141) hold by induction hypothesis and (59).

4 —— Iteration $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}$ where $\mathrm{at}_P\llbracket C\rrbracket = \ell$, $\mathrm{after}_P\llbracket C\rrbracket = \ell'$ and $\ell_1, \ell_2 \in \mathrm{in}_P\llbracket S\rrbracket$: According to the definition (130) of $\mathrm{APost}\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od},\rrbracket$, we start by the calculation of

$$
\begin{array}{lll}
& \mathrm{APost}^R\llbracket C\rrbracket(\ddot{\bot}[\ell \leftarrow r]) & \\
= & \quad \wr\text{def. (130) of } \mathrm{APost}^R\llbracket C\rrbracket\wr & \\
& \ddot{\bot}[\ell \leftarrow (\ddot{\bot}[\ell \leftarrow r])_{\mathrm{after}_P\llbracket S\rrbracket}] & \\
= & \quad \wr\text{labelling conditions (56) so that } \ell \neq \ell' \text{ and def. (16) of substitution}\wr & \\
& \ddot{\bot}[\ell \leftarrow \dot{\bot}] \quad = \quad \ddot{\bot}\ . & (142)
\end{array}
$$

It follows that

$$
\begin{array}{lll}
= & \Big(1_{\mathrm{ADom}\llbracket P\rrbracket \xrightarrow{\mathrm{mon}} \mathrm{ADom}\llbracket P\rrbracket} \mathbin{\dot{\ddot{\sqcup}}} (\mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^R\llbracket C\rrbracket)\Big)\,(\ddot{\bot}[\ell \leftarrow r]) & \\
& \quad \wr\text{(def. identity 1 and pointwise lub } \dot{\ddot{\sqcup}}\wr & \\
= & \big(\ddot{\bot}[\ell \leftarrow r] \mathbin{\ddot{\sqcup}} \mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^R\llbracket C\rrbracket(\ddot{\bot}[\ell \leftarrow r])\big) & \\
& \quad \wr\text{(142), strictness (139) and } \ddot{\bot} \text{ is the infimum}\wr & \\
= & \ddot{\bot}[\ell \leftarrow r]\ . & (143)
\end{array}
$$

For the fixpoint

$$
\begin{array}{ll}
& \Big(\lambda J\bullet \mathrm{lfp}^{\ddot{\sqsubseteq}}\ \lambda X\bullet J \mathbin{\ddot{\sqcup}} \mathrm{APost}^R\llbracket C\rrbracket \circ \mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^B\llbracket C\rrbracket(X)\Big)\,(\ddot{\bot}[\ell \leftarrow r]) \\
= & \quad \wr\text{def. application}\wr \\
& \mathrm{lfp}^{\ddot{\sqsubseteq}}\ \lambda X\bullet(\ddot{\bot}[\ell \leftarrow r]) \mathbin{\ddot{\sqcup}} \mathrm{APost}^R\llbracket C\rrbracket \circ \mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^B\llbracket C\rrbracket(X)\ ,
\end{array}
$$

let us define $\alpha' = \lambda x\bullet \ddot{\bot}[\ell \leftarrow x]$ and $\gamma' = \lambda J\bullet J_\ell$ so that we have the Galois connection

$$
\langle \mathrm{AEnv}\llbracket P\rrbracket,\ \dot{\sqsubseteq}\rangle \xrightleftharpoons[\alpha']{\gamma'} \langle \mathrm{ADom}\llbracket P\rrbracket,\ \ddot{\sqsubseteq}\rangle\ .
$$

We have

$$
\begin{array}{ll}
& (\ddot{\bot}[\ell \leftarrow r]) \mathbin{\ddot{\sqcup}} \mathrm{APost}^R\llbracket C\rrbracket \circ \mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^B\llbracket C\rrbracket(\alpha'(x)) \\
= & \quad \wr\text{def. } \alpha'\wr \\
& (\ddot{\bot}[\ell \leftarrow r]) \mathbin{\ddot{\sqcup}} \mathrm{APost}^R\llbracket C\rrbracket \circ \mathrm{APost}\llbracket S\rrbracket \circ \mathrm{APost}^B\llbracket C\rrbracket(\ddot{\bot}[\ell \leftarrow x]) \\
= & \quad \wr\text{def. (130) of } \mathrm{APost}^B\llbracket C\rrbracket\wr \\
& (\ddot{\bot}[\ell \leftarrow r]) \mathbin{\ddot{\sqcup}} \mathrm{APost}^R\llbracket C\rrbracket \circ \mathrm{APost}\llbracket S\rrbracket \circ \ddot{\bot}[\mathrm{at}_P\llbracket S\rrbracket \leftarrow \mathrm{Abexp}\llbracket B\rrbracket x] \\
= & \quad \wr\text{def. (136) of APostEn, (137) of } \alpha^e \text{ and induction hypothesis}\wr
\end{array}
$$

$$
\begin{aligned}
& (\ddot{\perp}[\ell \leftarrow r]) \mathrel{\ddot{\sqcup}} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x) \\
= \quad & \wr \text{def. (130) of } \mathrm{APost}^R[\![C]\!] \wr \\
& (\ddot{\perp}[\ell \leftarrow r]) \mathrel{\ddot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]} \\
= \quad & \wr \text{def. pointwise union } \ddot{\sqcup} \text{ and (16) of substitution} \wr \\
& (\ddot{\perp}[\ell \leftarrow r \mathrel{\dot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]}]) \\
= \quad & \wr \text{def. } \alpha' \wr \\
& \alpha' \circ \lambda x \cdot r \mathrel{\dot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]}(x) \ ,
\end{aligned}
$$

so that by the fixpoint abstraction theorem 2, we get

$$
\begin{aligned}
& \mathrm{lfp}^{\ddot{\sqsubseteq}} \lambda X \cdot (\ddot{\perp}[\ell \leftarrow r]) \mathrel{\ddot{\sqcup}} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](X) \\
= \ & \alpha'(\mathrm{lfp}^{\dot{\sqsubseteq}} \lambda x \cdot r \mathrel{\dot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]}(x)) \\
= \quad & \wr \text{def. } \alpha' \wr \\
& \ddot{\perp}[\ell \leftarrow \mathrm{lfp}^{\dot{\sqsubseteq}} \lambda x \cdot r \mathrel{\dot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]}(x)] \ .
\end{aligned}
\tag{144}
$$

It remains to calculate

$$
\begin{aligned}
& \left(1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\text{mon}} \mathrm{ADom}[\![P]\!]} \mathrel{\dot{\ddot{\sqcup}}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]) \mathrel{\dot{\ddot{\sqcup}}} \mathrm{APost}^{\bar{B}}[\![C]\!]\right) (\ddot{\perp}[\ell \leftarrow r']) \\
& \quad \wr \text{def. pointwise lub } \dot{\ddot{\sqcup}} \text{ and identity } 1 \wr \\
& \left(\ddot{\perp}[\ell \leftarrow r'] \mathrel{\ddot{\sqcup}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](\ddot{\perp}[\ell \leftarrow r'])) \mathrel{\ddot{\sqcup}} \mathrm{APost}^{\bar{B}}[\![C]\!](\ddot{\perp}[\ell \leftarrow r'])\right) \\
& \quad \wr \text{def. (130) of } \mathrm{APost}^{\bar{B}}[\![C]\!] \text{ and } \mathrm{APost}^{\bar{B}}[\![C]\!] \wr \\
& (\ddot{\perp}[\ell \leftarrow r'] \mathrel{\ddot{\sqcup}} (\mathrm{APost}[\![S]\!](\ddot{\perp}[\mathrm{at}_P[\![S]\!] \leftarrow \mathrm{Abexp}[\![B]\!]r'])) \mathrel{\ddot{\sqcup}} \ddot{\perp}[\ell' \leftarrow \mathrm{Abexp}[\![T(\neg B)]\!]r']) \\
& \quad \wr \text{labelling condition (56), def. (16) of substitution, def. (136) of APostEn, (137) of } \alpha^e \\
& \quad\ \ \text{and induction hypothesis} \wr \\
& \left(\ddot{\perp}[\ell \leftarrow r'; \ell' \leftarrow \mathrm{Abexp}[\![T(\neg B)]\!]r'] \mathrel{\ddot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]r')\right) \ .
\end{aligned}
\tag{145}
$$

It follows that for the iteration

$$
\begin{aligned}
& \mathrm{APostEn}_P[\![C]\!], \quad \text{where} \quad C = \texttt{while } B \texttt{ do } S \texttt{ od} \\
= \quad & \wr \text{def. (136) of APostEn and (137) of } \alpha^e \wr \\
& \lambda r \cdot \mathrm{APost}[\![C]\!](\ddot{\perp}[\ell \leftarrow r]) \\
= \quad & \wr \text{def. (130) of } \mathrm{APost}[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!] \wr \\
& \lambda r \cdot \left(1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\text{mon}} \mathrm{ADom}[\![P]\!]} \mathrel{\dot{\ddot{\sqcup}}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!]) \mathrel{\dot{\ddot{\sqcup}}} \mathrm{APost}^{\bar{B}}[\![C]\!]\right) \circ \\
& \qquad \left(\lambda J \cdot \mathrm{lfp}^{\ddot{\sqsubseteq}} \lambda X \cdot J \mathrel{\ddot{\sqcup}} \mathrm{APost}^R[\![C]\!] \circ \mathrm{APost}[\![S]\!] \circ \mathrm{APost}^B[\![C]\!](X)\right) \circ \\
& \qquad\quad \left(1_{\mathrm{ADom}[\![P]\!] \xrightarrow{\text{mon}} \mathrm{ADom}[\![P]\!]} \mathrel{\dot{\ddot{\sqcup}}} (\mathrm{APost}[\![S]\!] \circ \mathrm{APost}^R[\![C]\!])\right) (\ddot{\perp}[\ell \leftarrow r]) \\
= \quad & \wr \text{lemmata (143), (144) and (145)} \wr \\
& \lambda r \cdot \text{let } r' = \mathrm{lfp}^{\dot{\sqsubseteq}} \lambda x \cdot r \mathrel{\dot{\sqcup}} (\mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]x))_{\mathrm{after}_P[\![S]\!]}(x) \text{ in} \\
& \qquad (\ddot{\perp}[\ell \leftarrow r'; \ell' \leftarrow \mathrm{Abexp}[\![T(\neg B)]\!]r'] \mathrel{\ddot{\sqcup}} \mathrm{APostEn}_P[\![S]\!](\mathrm{Abexp}[\![B]\!]r')) \ .
\end{aligned}
$$

(140) and (141) hold by induction hypothesis, induction on the fixpoint iterates and (60).

5 — For the sequence $C_1 \ ; \ \ldots \ ; \ C_n, n > 0$ where $\ell = \mathrm{at}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] = \mathrm{at}_P[\![C_1]\!]$ and $\ell' = \mathrm{after}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] = \mathrm{after}_P[\![C_n]\!]$, we show that

$$
\begin{aligned}
\mathrm{APostEn}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!]r \quad = \quad & \text{let } J^1 = \mathrm{APostEn}_P[\![C_1]\!]r \text{ in} \\
& \quad \text{let } J^2 = J^1 \mathrel{\ddot{\sqcup}} \mathrm{APostEn}_P[\![C_2]\!] J^1_{\mathrm{at}_P[\![C_2]\!]} \text{ in} \\
& \quad\quad \ldots \\
& \quad\quad\quad \text{let } J^n = J^{n-1} \mathrel{\ddot{\sqcup}} \mathrm{APostEn}_P[\![C_n]\!](J^{n-1})_{\mathrm{at}_P[\![C_n]\!]} \text{ in} \\
& \quad\quad\quad\quad J^n \ ,
\end{aligned}
\tag{146}
$$

as well as the locality (140) and extension (141) properties. We proceed by induction on $n > 0$. This is trivial for the basis $n = 1$. For the induction step $n+1$, we have

$$
\begin{array}{ll}
& \mathrm{APostEn}_P[\![C_1 \ ; \ \ldots \ ; \ C_n \ ; \ C_{n+1}]\!] \\
= & \quad \wr \text{def. (136) of APostEn and (137) of } \alpha^e \wr \\
& \lambda r \bullet \mathrm{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n \ ; \ C_{n+1}]\!](\ddot{\bot}[\ell \leftarrow r]) \\
= & \quad \wr \text{def. (131) of } \mathrm{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n \ ; \ C_{n+1}]\!] \text{ and } \mathrm{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] \text{ and associa-} \\
& \quad \ \text{tivity of } \circ \wr \\
& \lambda r \bullet \mathrm{APost}[\![C_{n+1}]\!] \circ \mathrm{APost}[\![C_1 \ ; \ \ldots \ ; \ C_n]\!](\ddot{\bot}[\ell \leftarrow r]) \\
= & \quad \wr \text{def. (136) of APostEn, (137) of } \alpha^e \text{ and induction hypothesis (146)} \wr \\
& \lambda r \bullet \text{let } J^1 = \mathrm{APostEn}_P[\![C_1]\!]r \text{ in} \\
& \qquad \ldots \\
& \qquad \ \text{let } J^n = J^{n-1} \ddot{\sqcup} \mathrm{APostEn}_P[\![C_n]\!](J^{n-1})_{\mathrm{at}_P[\![C_n]\!]} \text{ in} \\
& \qquad \quad \mathrm{APost}[\![C_{n+1}]\!] J^n
\end{array}
$$

To conclude the induction step, it remains to calculate

$$
\begin{array}{ll}
& \mathrm{APost}[\![C_{n+1}]\!] J^n \\
= & \quad \wr \text{locality property (140), labelling (58) so that } \mathrm{after}_P[\![C_n]\!] = \mathrm{at}_P[\![C_{n+1}]\!] = \mathrm{in}_P[\![C_n]\!] \cap \\
& \quad \ \mathrm{in}_P[\![C_{n+1}]\!] \text{, locality (113) and dependence (114) properties} \wr \\
& \lambda l \in \mathrm{in}_P[\![C]\!] \bullet (l \in \mathrm{in}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] - \{\mathrm{at}_P[\![C_{n=1}]\!]\} \ ? \ (J^n)_l \\
& \qquad\qquad \text{¿ } \mathrm{APost}[\![C_{n+1}]\!] \ddot{\bot}[\mathrm{at}_P[\![C_{n=1}]\!] \leftarrow (J^n)_{\mathrm{at}_P[\![C_{n=1}]\!]}]) \\
= & \quad \wr \text{def. (136) of APostEn, (137) of } \alpha^e \text{ and structural induction} \wr \\
& \lambda l \in \mathrm{in}_P[\![C]\!] \bullet (l \in \mathrm{in}_P[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] - \{\mathrm{at}_P[\![C_{n=1}]\!]\} \ ? \ (J^n)_l \\
& \qquad\qquad \text{¿ } \mathrm{APostEn}_P[\![C_{n+1}]\!](J^n)_{\mathrm{at}_P[\![C_{n=1}]\!]}) \\
= & \quad \wr \text{locality (140) and extension (141)} \wr \\
& J^n \ddot{\sqcup} \mathrm{APostEn}_P[\![C_{n+1}]\!](J^n)_{\mathrm{at}_P[\![C_{n=1}]\!]} \ .
\end{array}
$$

so that

$$
\begin{array}{ll}
& \mathrm{APostEn}_P[\![C_1 \ ; \ \ldots \ ; \ C_{n+1}]\!]r \\
= & \text{let } J^1 = \mathrm{APostEn}_P[\![C_1]\!]r \text{ in} \\
& \quad \ldots \\
& \quad \ \text{let } J^{n+1} = J^n \ddot{\sqcup} \mathrm{APostEn}_P[\![C_{n+1}]\!](J^{n+1})_{\mathrm{at}_P[\![C_{n+1}]\!]} \text{ in} \\
& \qquad J^{n+1} \ .
\end{array}
$$

(140) and (141) hold by induction hypothesis and (58).

6 —— Programs $P = S$ ;;

$$
\begin{array}{ll}
& \mathrm{APostEn}_P[\![S \ ;;]\!] \\
= & \quad \wr \text{def. (136) of APostEn and (137) of } \alpha^e \wr \\
& \lambda r \bullet \mathrm{APost}[\![S \ ;;]\!](\ddot{\bot}[\ell \leftarrow r]) \\
= & \quad \wr \text{def. (132) of } \mathrm{APost}[\![S \ ;;]\!] \wr \\
& \lambda r \bullet \mathrm{APost}[\![S]\!](\ddot{\bot}[\ell \leftarrow r]) \\
= & \quad \wr \text{def. (136) of APostEn and (137) of } \alpha^e \wr \\
& \mathrm{APostEn}_P[\![S]\!] \ .
\end{array}
$$

The final specification is given in Fig. 16, from which programming is immediate [7]. Notice that the above calculation can be done directly on the program by partial evaluation [25] (although the present state of the art might not allow for a full automation of the program generation). The next step consists in avoiding useless copies of abstract invariants (deforestation).

$$
\begin{array}{ll}
\bullet & \mathrm{APostEn}_P\llbracket \texttt{skip} \rrbracket r \;=\; \ddot{\bot}[\mathrm{at}_P\llbracket \texttt{skip} \rrbracket \leftarrow r;\, \mathrm{after}_P\llbracket \texttt{skip} \rrbracket \leftarrow r] \\[1ex]
\bullet & \mathrm{APostEn}_P\llbracket \mathtt{X} := A \rrbracket r \;=\; \text{let } v = \mathrm{Faexp}^{\triangleright}\llbracket A \rrbracket r \text{ in} \\
& \quad (\mho(v) \;\mathbf{?}\; \ddot{\bot}[\mathrm{at}_P\llbracket \mathtt{X} := A \rrbracket \leftarrow r] \\
& \quad\quad \text{¿}\; \ddot{\bot}[\mathrm{at}_P\llbracket \mathtt{X} := A \rrbracket \leftarrow r;\, \mathrm{after}_P\llbracket \mathtt{X} := A \rrbracket \leftarrow r[\mathtt{X} \leftarrow v \sqcap ?^{\triangleright}]]) \\[1ex]
\bullet & C = \texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}, \quad \mathrm{APostEn}_P\llbracket C \rrbracket r \;= \\
& \quad \text{let } J^{\mathrm{tt}} = \mathrm{APostEn}_P\llbracket S_t \rrbracket(\mathrm{Abexp}\llbracket B \rrbracket r) \text{ in} \\
& \quad\quad \text{let } J^{\mathrm{ff}} = \mathrm{APostEn}_P\llbracket S_f \rrbracket(\mathrm{Abexp}\llbracket T(\neg B) \rrbracket r) \text{ in} \\
& \quad\quad\quad \ddot{\bot}[\mathrm{at}_P\llbracket C \rrbracket \leftarrow r;\, \mathrm{after}_P\llbracket C \rrbracket \leftarrow J^{\mathrm{tt}}_{\mathrm{after}_P\llbracket S_t \rrbracket} \mathbin{\dot{\sqcup}} J^{\mathrm{ff}}_{\mathrm{after}_P\llbracket S_f \rrbracket}] \mathbin{\ddot{\sqcup}} J^{\mathrm{tt}} \mathbin{\ddot{\sqcup}} J^{\mathrm{ff}} \\[1ex]
\bullet & C = \texttt{while } B \texttt{ do } S \texttt{ od}, \quad \mathrm{APostEn}_P\llbracket C \rrbracket r \;= \\
& \quad \text{let } r' = \mathrm{lfp}^{\dot{\sqsubseteq}}\, \lambda x \bullet r \mathbin{\dot{\sqcup}} (\mathrm{APostEn}_P\llbracket S \rrbracket(\mathrm{Abexp}\llbracket B \rrbracket x))_{\mathrm{after}_P\llbracket S \rrbracket} \text{ in} \\
& \quad\quad \ddot{\bot}[\mathrm{at}_P\llbracket C \rrbracket \leftarrow r';\, \mathrm{after}_P\llbracket C \rrbracket \leftarrow \mathrm{Abexp}\llbracket T(\neg B) \rrbracket r'] \mathbin{\ddot{\sqcup}} \\
& \quad\quad\quad \mathrm{APostEn}_P\llbracket S \rrbracket(\mathrm{Abexp}\llbracket B \rrbracket r') \\[1ex]
\bullet & C = C_1 \;\texttt{;}\; \ldots \;\texttt{;}\; C_n, \quad \mathrm{APostEn}_P\llbracket C \rrbracket r \;= \\
& \quad \text{let } J^1 = \mathrm{APostEn}_P\llbracket C_1 \rrbracket r \text{ in} \\
& \quad\quad \text{let } J^2 = J^1 \mathbin{\ddot{\sqcup}} \mathrm{APostEn}_P\llbracket C_2 \rrbracket(J^1)_{\mathrm{at}_P\llbracket C_2 \rrbracket} \text{ in} \\
& \quad\quad\quad \ldots \\
& \quad\quad\quad\quad \text{let } J^n = J^{n-1} \mathbin{\ddot{\sqcup}} \mathrm{APostEn}_P\llbracket C_n \rrbracket(J^{n-1})_{\mathrm{at}_P\llbracket C_n \rrbracket} \text{ in} \\
& \quad\quad\quad\quad\quad J^n \\[1ex]
\bullet & \mathrm{APostEn}_P\llbracket S \;\texttt{;;} \rrbracket \;=\; \mathrm{APostEn}_P\llbracket S \rrbracket(\lambda \mathtt{X} \in \mathrm{Var}\llbracket P \rrbracket \bullet \alpha(\{\Omega_{\mathtt{i}}\}))\;.
\end{array}
\tag{147}
$$

Figure 16: Generic forward nonrelational reachability from entry states abstract semantics of programs

By choosing to totally order the labels (such that $\mathrm{in}_P\llbracket C \rrbracket = \{\ell \mid \mathrm{at}_P\llbracket C \rrbracket \leq \ell \leq \mathrm{after}_P\llbracket C \rrbracket\}$) and the program variables, abstract invariants can be efficiently represented as matrices of abstract values. The locality (140) and dependence (equivalent to (114)) properties for $\mathrm{APostEn}_P\llbracket C \rrbracket$ yield an implementation where the abstract invariant is computed by assignments to a global array. For large programs more efficient memory management strategies are necessary which is facilitated by the observation that the only global information needing to be permanently memorized are the loop abstract invariants.

## 14. Conclusion

These notes cover in part the 1998 Marktoberdorf course on the *“calculational design of semantics and static analyzers by abstract interpretation”*. We have chosen to put the emphasis on the calculational design more than on the abstract interpretation theory and its possible applications to software reliability. The objective of these notes is to show clearly that the complete calculation-based development of program analyzers is possible, which is much more difficult to explain orally.

The programming language considered in the course was the same, except that the small-step operational semantics (Sec. 7., 9. and 12.) was defined using an ML-type based abstract syntax (indeed isomorphic to the grammar based abstract syntax of Sec. 7.1, 9.1 and 12.1).

We considered a hierarchy of semantics by abstraction of a infinitary trace semantics expressed in fixpoint form (see [6]). The non-classical big-step operational semantics of Sec. 12.8 and reachable states semantics of Sec. 12.10 are only two particular semantics in this rich hierarchy. The interest of this point of view is to rub out the dependence upon the particular standard semantics which is used as initial specification of program behaviors since all semantics are abstract interpretations of one another, hence are all part of the same lattice of abstract interpretations [13].

The Galois connection and widening/narrowing based abstract interpretation frameworks (including the combination and refinement of abstract algebras) were treated at length. Very few elements are given in these written notes (see complements in [14, 16, 17] at higher-order and [15] for weaker frameworks not assuming the existence of best approximations). Finite abstract algebras like the initialization and simple sign domain of Sec. 5.3 are often not expressive enough in practice. One must resort to infinite abstract domains like the intervals considered in the course (see [8, 9]), which is the smallest abstract domain which is *complete* for determining the sign of addition [23]. With such infinite abstract domains which do not satisfy the ascending chain condition, widening/narrowing are needed for accelerating the convergence and improving the precision of fixpoint computations.

Being based on a particular abstract syntax and semantics, the recursive analyzer considered in these notes is dependent upon the language to be analyzed. This was avoided in the course since the design of generic abstract interpreters was based on compositionally defined systems of equations, chaotic iterations and weak topological orderings.

The emphasis in these notes has been on the correctness of the design by calculus. The mechanized verification of this formal development using a proof assistant can be foreseen with automatic extraction of a correct program from its correctness proof [30]. Unfortunately most proof assistants are presently still unstable, heavy if not rebarbative to use and sometimes simply bugged.

The specification of the static analyzer which has been derived in these course notes is well-adapted to the higher-order modular functional programming style. Further refinement steps would be necessary for efficiency. The problem of deriving very efficient analyzers which are both fast and memory sparing goes beyond classical compiler program optimization and partial evaluation techniques (as shown by the specialization to entry states in Sec. 13.10). This problem has not been considered in the course nor in these notes.

A balance between correctness and efficiency might be found by developing both an efficient static analyzer (with expensive fixpoint computations, etc.) and a correct static verifier (which might be somewhat inefficient to perform a mere checking of the abstract invariant computed by the analyzer). Only the correctness of the verifier must be formally established without particular concern for efficiency.

The main application of the program static analyzer considered in the course was *abstract checking*, as introduced[10] in [5] and refined by [2]. The difference with abstract model-checking [19] is that the semantic model is not assumed to be finite, the abstraction is not specific to a particular program (see [16] for a proof that finite abstract domains are inadequate in this context) and specifications are not given using a temporal logic. By experience, specifications separated from the program do not evolve with program modifications over large periods of time (10 to 20 years) and are unreadable for very large programs (over 100,000 lines). The solution proposed in the oral course was to insert safety/invariant and liveness/intermittent

[10] without name

together with final and initial assertions in the program text. The analysis must then combine forward and backward abstract interpretations (only forward analyses were considered in these written notes, see e.g. [18] for this more general case and an explanation of why decreasing iterations are necessary in the context of infinite systems).

The final question is whether the calculational design of program static analyzers by abstract interpretation of a formal semantics does scale up. Experience shows that it does by small parts. This provides a thorough understanding of the abstraction process allowing for the later development of useful large scale analyzers [27].


## Acknowledgments

I thank Manfred Broy and the organizers of the International Summer School Marktoberdorf (Germany) on “Calculational System Design” for inviting me to the course and to write these notes. I thank Radhia Cousot and Roberto Giacobazzi for their comments on a draft.


## References


[1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers. Principles, Technique and Tools*. Addison-Wesley, 1986. 19

[2] F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Proc. PLDI*, pp. 46–55. ACM Press, 1993. 85

[3] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In D. Bjørner, M. Broy, and I.V. Pottosin, editors, *Proc. FMPA*, Academgorodok, Novosibirsk, Russia, LNCS 735, pp. 128–141. Springer, Jun. 28–Jul. 2, 1993. 59

[4] P. Cousot. *Méthodes itératives de construction et d’approximation de points fixes d’opérateurs monotones sur un treillis, analyse sémantique de programmes*. Thèse d’État ès sciences mathématiques, Université scientifique et médicale de Grenoble, France, 21 March 1978. 59, 59

[5] P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, eds., *Program Flow Analysis: Theory and Applications*, ch. 10, pp. 303–342. Prentice-Hall, 1981. 3, 59, 59, 59, 85

[6] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *ENTCS*, 6, 1997. URL: `http://www.elsevier.nl/locate/entcs/volume6.html`, 25 pages. 3, 85

[7] P. Cousot. The Marktoberdorf’98 generic abstract interpreter. Available at URL: `http://www.dmi.ens.fr/~cousot/Marktoberdorf98.shtml`. 19, 75, 77, 79, 83

[8] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. 2nd Int. Symp. on Programming*, pp. 106–130. Dunod, 1976. 8, 85

[9] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL*, pp. 238–252, Los Angeles, Calif., 1977. ACM Press. 3, 3, 8, 39, 74, 85

[10] P. Cousot and R. Cousot. Automatic synthesis of optimal invariant assertions: mathematical foundations. In *ACM Symposium on Artificial Intelligence & Programming Languages*, Rochester, N.Y., SIGPLAN Notices 12(8):1–12, 1977. 59

[11] P. Cousot and R. Cousot. A constructive characterization of the lattices of all retractions, preclosure, quasi-closure and closure operators on a complete lattice. *Portugal. Math.*, 38(2):185–198, 1979. 38

[12] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific J. Math.*, 82(1):43–57, 1979. 38, 60, 60, 62

[13] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6$^{th}$ POPL*, pp. 269–282, San Antonio, Texas, 1979. ACM Press. 3, 6, 11, 15, 15, 15, 37, 59, 59, 59, 60, 61, 85

[14] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 13(2–3):103–179, 1992. (The editor of JLP has mistakenly published the unreadable galley proof. For a correct version of this paper, see `http://www.dmi.ens.fr/~cousot`.). 7, 11, 85

[15] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, Aug. 1992. 85

[16] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing, editors, *Proc. Int. Work. PLILP '92,*, Leuven, Belgium, LNCS 631, pp. 269–295. Springer, 13–17 Aug. 1992. 6, 6, 74, 85, 85

[17] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *Proc. 1994 ICCL*, Toulouse, France, pp. 95–112. IEEE Comp. Soc. Press, 16–19 May 1994. 85

[18] P. Cousot and R. Cousot. Refining model checking by abstract interpretation. *Automated Software Engineering Journal, special issue on Automated Software Analysis*, 6(1), 1999. To appear. 86

[19] D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL$^\star$, $\exists$CTL$^\star$ and CTL$^\star$. In E.R. Olderog, editor, *Proc. IFIP WG2.1/WG2.2/WG2.3 Working Conf. on Programming Concepts, Methods and Calculi (PROCOMET)*, IFIP Transactions. North-Holland/Elsevier, Jun. 1994. 85

[20] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge U. Press, 1990. 4

[21] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, ch. 6, pp. 243–320. Elsevier, 1990. 41

[22] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics*. Springer, 1990. 14

[23] R. Giacobazzi and F. Ranzato. Completeness in abstract interpretation: A domain perspective. In M. Johnson, editor, *Proc. 6$^{th}$ Int. Conf. AMAST '97, Sydney, Australia*, LNCS 1349, pp. 231–245. Springer, 13–18 Dec. 1997. 85

[24] P. Granger. Improving the results of static analyses of programs by local decreasing iterations. In *Proc. 12$^{th}$ FST & TCS*, pp. 68–79, New Delhi, India, LNCS 652. Springer, 18–20 Dec. 1992. 38

[25] N.D. Jones, Gomard C.K., Sestoft P., L.O. (Andersen, and T.) Mogensen. *Partial Evaluation and Automatic Program Generation*. Prentice-Hall, 1993. 83

[26] G. Kahn. Natural semantics. In K. Fuchi and M. Nivat, editors, *Programming of Future Generation Computers*, pp. 237–258. Elsevier, 1988. 13

[27] P. Lacan, J.N. Monfort, Le Vinh Quy Ribal, A. Deutsch, and G. Gonthier. The software reliability verification process: The ARIANE 5 example. In *Proc. DASIA 98 – DAta Systems IN Aerospace*, Athens, Grece. ESA Publications, SP-422, May 25–28 1998. 3, 86

[28] B. Le Charlier and P. Flener. On the desirable link between theory and practice in abstract interpretation. In P. Van Hentenryck, ed., *Proc. SAS '97*, Paris, FRA, LNCS 1302, pp. 379–387. Springer, 8–10 Sep. 1997. 15, 15

[29] B. Le Charlier and P. Van Hentenryck. Reexecution in Abstract Interpretation of Prolog. In Krzysztof Apt, ed., *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pp. 750–764, Washington, USA, Nov. 1992. The MIT Press. 38

[30] D. Monniaux. Réalisation mécanisée d'interpréteurs abstraits. Rapport de stage, DEA "Sémantique, Preuve et Programmation", Jul. 1998. 85

[31] G.D. Plotkin. A structural approach to operational semantics. Tech. rep. DAIMI FN-19, Aarhus University, Denmark, Sep. 1981. 13, 31, 43, 57

[32] E. Schön. On the computation of fixpoints in static program analysis with an application to analysis of AKL. Res. rep. R95:06, Swedish Institute of Computer Science, SICS, 1995. 59