University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master degree in Computer Science

Titolo della tesi

Supervisor
Prof. Prof 1

Co. Supervisor
Prof. Prof 2

Candidate
Luca Zaninotto

Academic Year 2023-2024

Abstract

Abstract

Acknowledgments

?

Contents

1 Framework **1**

1.1 The Imp language 1

1.2 Semantics 1

1.2.1 Syntactic sugar 2

1.2.2 Small step semantics 2

1.3 Transition system 4

1.4 Functions in Imp 6

1.5 Deciding invariant finiteness 9

2 Intervals **13**

2.1 Interval Analysis 13

2.2 Computing the interval semantics 19

3 Non relational collecting **29**

Chapter 1

Framework

1.1 The Imp language

In order to talk about program properties we need a language to express such programs. We define the Imp language, made of regular commands and based on Kozen's Kleene algebra with tests, described in [Koz97]. We denote by $\mathbb{Z}$ the set of integers with the usual order, extended with bottom and top elements $-\infty$ and $+\infty$, s.t. $-\infty \leqslant z \leqslant +\infty \quad \forall z \in \mathbb{Z}$. We also extend addition and subtraction by letting, for $z \in \mathbb{Z} \quad +\infty + z = +\infty - z = +\infty$ and $-\infty + z = -\infty - z = -\infty$. We focus on the following non-deterministic language.

$$
\begin{aligned}
\mathsf{Exp} \ni \mathsf{e} &::= \mathtt{x} \in S \mid \mathsf{true} \mid \mathsf{false} \mid \mathtt{x} := k \mid \mathtt{x} := \mathtt{y} + k \\
\mathrm{Imp}_{\neq\star} \ni \mathsf{D} &::= \mathsf{e} \mid \mathsf{D} + \mathsf{D} \mid \mathsf{D}; \mathsf{D} \\
\mathrm{Imp} \ni \mathsf{C} &::= \mathsf{D} \mid \mathsf{C} + \mathsf{C} \mid \mathsf{C}; \mathsf{C} \mid \mathsf{C}^* \mid \mathsf{fix}(\mathsf{C})
\end{aligned}
$$

where $\mathtt{x}, \mathtt{y} \in \mathit{Var}$ a finite set of variables of interest, i.e., the variables appearing in the considered program, $S \subseteq \mathbb{Z}$ is (possibly empty) *decidable* set of numbers, $a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}, a \leqslant b, k \in \mathbb{Z}$ is any finite integer constant.

1.2 Semantics

In order to talk about program properties in our language, we first need to define its *semantics*. In the following section we introduce both a collecting semantics in order to reason about program *invariants* and a small step semantics, in order to reason about program *execution*.

Definition 1.1 (Semantics of Basic Expressions)**.** Let *environments* be the maps from the set of variables to their numerical value: $\mathsf{Env} \triangleq \{\rho \mid \rho : \mathit{Var} \to \mathbb{Z}\}$. For basic expressions $e \in \mathsf{Exp}$ the *concrete semantics* $(\!|\cdot|\!) : \mathsf{Exp} \to \mathsf{Env} \to \mathsf{Env}_\bot$ is inductively defined by:

$$
\begin{aligned}
(\!|\mathtt{x} \in S|\!)\rho &\triangleq \begin{cases} \rho & \rho(\mathtt{x}) \in S \\ \bot & \text{otherwise} \end{cases} \\
(\!|\mathsf{true}|\!)\rho &\triangleq \rho \\
(\!|\mathsf{false}|\!)\rho &\triangleq \bot \\
(\!|\mathtt{x} := k|\!)\rho &\triangleq \rho[\mathtt{x} \mapsto k] \\
(\!|\mathtt{x} := \mathtt{y} + k|\!)\rho &\triangleq \begin{cases} \rho[\mathtt{x} \mapsto \rho(\mathtt{y}) + k] & \rho \neq \bot \\ \bot & \text{otherwise} \end{cases}
\end{aligned}
$$

The next building block is the concrete collecting semantics for the language, it associates each program in Imp to a function which, given a set of initial environments X "collects" the set of final states produced by executing the program from X.

Definition 1.2 (Concrete collecting semantics)**.** Let $\mathbb{C} \triangleq \langle 2^{\mathsf{Env}}, \subseteq \rangle$ be a complete lattice called *concrete collecting domain.* The *concrete collecting semantics* for Imp is given by the total function $\langle \cdot \rangle : \text{Imp} \to \mathbb{C} \to \mathbb{C}$ which maps each program $\mathsf{C} \in \text{Imp}$ to a total function over the complete lattice $\mathbb{C}$, inductively defined as follows: given $X \in \mathbb{C}$

$$\begin{aligned}
\langle \mathsf{e} \rangle X &\triangleq \{ (\!|\mathsf{e}|\!)\rho \mid \rho \in X, (\!|\mathsf{e}|\!)\rho \neq \bot \} \\
\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle X &\triangleq \langle \mathsf{C}_1 \rangle X \cup \langle \mathsf{C}_2 \rangle X \\
\langle \mathsf{C}_1 ; \mathsf{C}_2 \rangle X &\triangleq \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X) \\
\langle \mathsf{C}^* \rangle X &\triangleq \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i X \\
\langle \mathsf{fix}(\mathsf{C}) \rangle X &\triangleq \text{lfp}(\lambda Y \in 2^{\mathsf{Env}}.(X \cup Y))
\end{aligned}$$

This concrete semantics is additive, this implies that the Kleene star (C^*) and the fixpoint $(\mathsf{fix}(\mathsf{C}))$ have the same concrete semantics $\langle \mathsf{C}^* \rangle = \langle \mathsf{fix}(\mathsf{C}) \rangle$. This will not be the case for the abstract semantics (cf. example 2.1), where the Kleene star can be more precise than the fixpoint semantics, but harder to compute and, as such, less suited for analysis. For the concrete semantics, however, since they are the same in the next proofs we only explore the case C^* since it captures also $\mathsf{fix}(\mathsf{C})$. Since for a given program C and a set of initial states $X \in \mathbb{C}$ the collecting semantics $\langle \mathsf{C} \rangle X$ expresses properties that hold at the end of the execution of C we will in the following chapters usually refer to $\langle \mathsf{C} \rangle X$ as program *invariant.*

Notation 1.1 (Singleton shorthand)**.** Sometimes we need to consider the semantics over the singleton set $\{\rho\}$. In these cases we will write $\langle \mathsf{C} \rangle \rho$ meaning $\langle \mathsf{C} \rangle \{\rho\}$.

1.2.1 Syntactic sugar

We define some syntactic sugar for the language. In the next chapters we will often use the syntactic sugar instead of its real equivalent for the sake of simplicity.

$$\begin{aligned}
\mathtt{x} \in [a,b] &= \mathtt{x} \in S && \text{with } S = [a,b], \text{ decidable} \\
\mathtt{x} \leqslant k &= \mathtt{x} \in (-\infty, k] \\
\mathtt{x} > k &= \mathtt{x} \in [k+1, +\infty) \\
\mathtt{x} \in S_1 \vee \mathtt{x} \in S_2 &= (\mathtt{x} \in S_1) + (\mathtt{x} \in S_2) \\
\mathtt{x} \in S_1 \wedge \mathtt{x} \in S_2 &= (\mathtt{x} \in S_1) ; (\mathtt{x} \in S_2) \\
\mathtt{x} \notin S &= \mathtt{x} \in \neg S \\
\mathsf{if}\ b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 &= (\mathsf{e}; \mathsf{C}_1) + (\neg \mathsf{e}; \mathsf{C}_2) \\
\mathsf{while}\ b\ \mathsf{do}\ C &= \mathsf{fix}(\mathsf{e}; \mathsf{C}); \neg \mathsf{e} \\
\mathtt{x++} &= \mathtt{x} := \mathtt{x} + 1
\end{aligned}$$

1.2.2 Small step semantics

Now that we have defined the collecting semantics to express program properties, we need the small step semantics to talk about program execution. We start by defining *program states*: $\mathsf{State} \triangleq \text{Imp} \times \mathsf{Env}$ tuples of programs and program environments. With states we can define our small step semantics:

Definition 1.3 (Small step semantics)**.** The small step transition relation for the language Imp $\to$: $\mathsf{State} \times (\mathsf{State} \cup \mathsf{Env})$ is defined by the following rules:

$$\frac{(\!|e|\!)\rho \neq \bot}{\langle \mathsf{e}, \rho \rangle \to (\!|e|\!)\rho}\ \text{expr}$$

$$\frac{}{\langle \mathsf{C}_1 + \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_1, \rho\rangle}\ \text{sum}_1 \qquad \frac{}{\langle \mathsf{C}_1 + \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_2, \rho\rangle}\ \text{sum}_2$$

$$\frac{\langle \mathsf{C}_1, \rho\rangle \to \langle \mathsf{C}_1', \rho'\rangle}{\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_1'; \mathsf{C}_2, \rho'\rangle}\ \text{comp}_1 \qquad \frac{\langle \mathsf{C}_1, \rho\rangle \to \rho'}{\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_2, \rho'\rangle}\ \text{comp}_2$$

$$\frac{}{\langle \mathsf{C}^*, \rho\rangle \to \langle \mathsf{C}; \mathsf{C}^*, \rho\rangle}\ \text{star} \qquad \frac{}{\langle \mathsf{C}^*, \rho\rangle \to \rho}\ \text{star}_{\text{fix}}$$

With the following lemma we introduce a link between the small step semantics and the concrete collecting semantics: the invariant of a program is the collection of all the environments the program halts on when executing.

Lemma 1.1. *For any* $\mathsf{C} \in \mathit{Imp}, X \in 2^{\mathsf{Env}}$

$$\langle \mathsf{C}\rangle X = \{\rho' \in \mathsf{Env} \mid \rho \in X, \langle \mathsf{C}, \rho\rangle \to^* \rho'\}$$

where $\to^*$ is the reflexive and transitive closure of the $\to$ relation.

Proof. by induction on C:

Base case:
$\mathsf{C} \equiv \mathsf{e}$
$\langle \mathsf{e}\rangle X = \{(\!|e|\!)\rho \mid \rho \in X \wedge (\!|e|\!)\rho \neq \bot\}$, $\forall \rho \in X.\langle \mathsf{e}, \rho\rangle \to (\!|e|\!)\rho$ if $(\!|e|\!)\rho \neq \bot$, and because of the expr rule

$$\langle \mathsf{e}\rangle X = \{(\!|e|\!)\rho \mid \rho \in X \wedge (\!|e|\!)\rho \neq \bot\} = \{\rho' \in \mathsf{Env} \mid \rho \in X\langle \mathsf{e}, \rho\rangle \to \rho'\}$$

Inductive cases:

- $\mathsf{C} \equiv \mathsf{C}_1 + \mathsf{C}_2$
  $\langle \mathsf{C}_1 + \mathsf{C}_2\rangle X = \langle \mathsf{C}_1\rangle X \cup \langle \mathsf{C}_2\rangle X$, $\forall \rho \in X.\langle \mathsf{C}_1 + \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_1, \rho\rangle \vee \langle \mathsf{C}_1 + \mathsf{C}_2, \rho\rangle \to \langle \mathsf{C}_2, \rho\rangle$ respectively according to rules sum$_1$ and sum$_2$. By inductive hypothesis

  $$\langle \mathsf{C}_1\rangle X = \{\rho' \in \mathsf{Env} \mid \rho \in X, \langle \mathsf{C}_1, \rho\rangle \to^* \rho'\} \quad \langle \mathsf{C}_2\rangle X = \{\rho' \in \mathsf{Env} \mid \rho \in X, \langle \mathsf{C}_2, \rho\rangle \to^* \rho'\}$$

  Therefore

  $$\begin{aligned}\langle \mathsf{C}_1 + \mathsf{C}_2\rangle X &= \langle \mathsf{C}_1\rangle X \cup \langle \mathsf{C}_2\rangle X && \text{(by definition)}\\ &= \{\rho' \in \mathsf{Env} \mid \rho \in X.\langle \mathsf{C}_1, \rho\rangle \to^* \rho'\} \cup \{\rho' \in \mathsf{Env} \mid \rho \in X, \langle \mathsf{C}_2, \rho\rangle \to^* \rho'\} && \text{(by ind. hp)}\\ &= \{\rho' \in \mathsf{Env} \mid \rho \in X.\langle \mathsf{C}_1, \rho\rangle \to^* \rho' \vee \langle \mathsf{C}_2, \rho\rangle \to^* \rho'\}\\ &= \{\rho' \in \mathsf{Env} \mid \rho \in X.\langle \mathsf{C}_1 + \mathsf{C}_2, \rho\rangle \to^* \rho'\}\end{aligned}$$

- $\mathsf{C} \equiv \mathsf{C}_1; \mathsf{C}_2$
  $\langle \mathsf{C}_1; \mathsf{C}_2\rangle X = \langle \mathsf{C}_2\rangle(\langle \mathsf{C}_1\rangle X)$. By inductive hp $\langle \mathsf{C}_1\rangle X = \{\rho' \in \mathsf{Env} \mid \rho \in X, \langle \mathsf{C}_1, \rho\rangle \to^* \rho'\} = Y$, by inductive hp again $\langle \mathsf{C}_2\rangle Y = \{\rho' \in \mathsf{Env} \mid \rho \in Y, \langle \mathsf{C}_2, \rho\rangle \to^* \rho'\}$. Therefore

  $$\begin{aligned}\langle \mathsf{C}_1; \mathsf{C}_2\rangle X &= \langle \mathsf{C}_2\rangle(\langle \mathsf{C}_1\rangle X) && \text{(by definition)}\\ &= \{\rho' \in \mathsf{Env} \mid \rho'' \in \{\rho''' \mid \rho \in X, \langle \mathsf{C}_1, \rho\rangle \to^* \rho'''\}, \langle \mathsf{C}_2, \rho''\rangle \to^* \rho'\} && \text{(by ind. hp)}\\ &= \{\rho' \in \mathsf{Env} \mid \rho \in X.\langle \mathsf{C}_1, \rho\rangle \to^* \rho'' \wedge \langle \mathsf{C}_2, \rho''\rangle \to^* \rho'\} && \text{(by composition lemma)}\\ &= \{\rho' \in \mathsf{Env} \mid \rho \in X.\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^* \rho'\}\end{aligned}$$

- $\mathsf{C} \equiv \mathsf{C}^*$
  $\langle \mathsf{C}^* \rangle X = \cup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i X$

$$\begin{aligned}
\langle \mathsf{C}^* \rangle X &= X \cup \langle \mathsf{C} \rangle X \cup \langle \mathsf{C} \rangle^2 X \cup \dots && \text{(by definition)} \\
&= X \cup \{\rho' \in \mathsf{Env} \mid \rho \in X. \langle \mathsf{C}, \rho \rangle \to^* \rho'\} \cup \dots && \text{(by ind. hp)} \\
&= \cup_{i \in \mathbb{N}} \{\rho' \in \mathsf{Env} \mid \rho \in X. \langle \mathsf{C}^i, \rho \rangle \to^* \rho'\} \\
&= \{\rho' \in \mathsf{Env} \mid \rho \in X. \vee_{i \in \mathbb{N}} \langle \mathsf{C}^i, \rho \rangle \to^* \rho'\} \\
&= \{\rho' \in \mathsf{Env} \mid \rho \in X. \langle \mathsf{C}^*, \rho \rangle \to^* \rho'\}
\end{aligned}$$

□

Notice that $\langle \mathsf{C} \rangle X = \varnothing \iff \nexists \rho' \in \mathsf{Env}, \rho \in X \mid \langle \mathsf{C}, \rho \rangle \to^* \rho'$, in other words the collecting semantics of some program C starting from some states $X \in \mathbb{C}$ is empty iff the program never halts on some state ρ'. Another observation is that due to non-determinism a program can halt on multiple final states, or have one branch of execution that halts on some final state, while the other never halts on any final state. Non-determinism implies that there are two different types of termination, intuitively a program can *always* halt or *partially* halt. We will better explore this concept in the next chapter.

1.3 Transition system

With the set of states State, the set of environments Env and the small operational semantics $\to$ we define a transition system, this will be useful to define universal and partial termination and to reason about program properties in the next chapters.

Definition 1.4 (Transition system)**.** The transition system for the language Imp is

$$\mathsf{TS} \triangleq \langle \mathsf{State} \cup \mathsf{Env}, \mathsf{Env}, \to \rangle$$

where

- $\mathsf{State} \cup \mathsf{Env}$ is the set of configurations in the system;
- Env is the set of terminal states;
- $\to$ is the small step semantics defined in definition 1.3, which describes the transition relations in the system.

In such a system we define *paths*, sequences of $\to$ relations starting from some state.

Definition 1.5 (Paths)**.** Let $(\mathsf{State} \cup \mathsf{Env})^\infty \triangleq (\mathsf{State} \cup \mathsf{Env})^+ \cup (\mathsf{State} \cup \mathsf{Env})^\omega$ be the set of all infinitary sequences of states and environments (both finite and infinite). Then the set of *paths* in the transition system is

$$\mathsf{Path}^\infty \triangleq \{\tau \in (\mathsf{State} \cup \mathsf{Env})^\infty \mid \forall i \in [1, |\tau|). \tau_i \to \tau_{i+1}\}$$

Where by X^+ we mean all the finite sequences of (possibly repeating) elements of X, while with X^ω we mean all the infinite sequences of (possibly repeating) elements of X.

In this context we are more interested in paths starting from some known state $\langle \mathsf{C}, \rho \rangle$ for some program $\mathsf{C} \in$ Imp and some initial state $\rho \in \mathsf{Env}$. We express this by overloading the $\langle \mathsf{C}, \rho \rangle$ notation:

Definition 1.6. Given $\mathsf{C} \in$ Imp$, \rho \in \mathsf{Env}$ the *paths* in the transition system starting from $\langle \mathsf{C}, \rho \rangle$ are

$$\langle \mathsf{C}, \rho \rangle \triangleq \{\tau \in \mathsf{Path}^\infty \mid \tau_0 = \langle \mathsf{C}, \rho \rangle\}$$

With the concept of paths we can define what we mean for *partial* and *universal* termination.

$$\begin{tikzcd}
\langle (\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0, \rho \rangle \arrow[r] \arrow[d] & \langle \mathtt{x} < 0, \rho \rangle \not\to \\
\langle \mathtt{x} \geq; 0\mathtt{x++}; (\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0, \rho \rangle \arrow[d, "*"] & \\
\langle (\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + 1] \rangle \arrow[r] \arrow[d, "*"] & \langle \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + 1] \rangle \not\to \\
\langle (\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + k] \rangle \arrow[r] & \langle \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + k] \rangle \not\to
\end{tikzcd}$$

Figure 1.1: Transition system of $(\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0$

Definition 1.7 (Partial termination)**.** Let $\mathsf{C} \in \text{Imp}, \rho \in \mathsf{Env}$. C *partially halts* on ρ when there's at least one path of finite length in the transition system $\langle \mathsf{C}, \rho \rangle$ ending up in some state ρ':

$$\langle \mathsf{C}, \rho \rangle \downarrow \iff \exists k \in \mathbb{N} \mid \langle \mathsf{C}, \rho \rangle \to^k \rho'.$$

Dually

$$\langle \mathsf{C}, \rho \rangle \upuparrows \iff \neg \langle \mathsf{C}, \rho \rangle \downarrow$$

a program *always loops* if there's no finite path in its transition system that leads to a final environment.

Definition 1.8 (Universal termination)**.** Let $\mathsf{C} \in \text{Imp}, \rho \in \mathsf{Env}$. C *partially loops* on ρ when there's at least one path of infinite length in the transition system $\langle \mathsf{C}, \rho \rangle$:

$$\langle \mathsf{C}, \rho \rangle \uparrow \iff \forall k \in \mathbb{N}\ \langle \mathsf{C}, \rho \rangle \to^k \langle \mathsf{C}', \rho' \rangle \quad \text{for some } \mathsf{C}' \in \text{Imp}, \rho' \in \mathsf{Env}.$$

Dually

$$\langle \mathsf{C}, \rho \rangle \downdownarrows \iff \neg \langle \mathsf{C}, \rho \rangle \uparrow$$

a program *universally halts* iff there's no infinite path in the transition systems.

Example 1.3 shows a program that partially halts, while example 1.2 shows a program that always loops. Notice that the absence of infinite paths implies that $\langle \mathsf{C}, \rho \rangle$ is finite. Example 1.3 shows a program that partially loops, while example 1.1 shows a program that universally halts.

Example 1.1. Consider the program

$$\mathtt{x} := 0;$$

always halts, since $\forall \rho \in \mathsf{Env}, \rho \neq \bot$ builds the transition system

$$\langle \mathtt{x} := 0, \rho \rangle \to \rho[\mathtt{x} \mapsto 0]$$

according to the expr rule in definition 1.3. Therefore $\langle (\mathtt{x} := 0), \rho \rangle \downdownarrows \forall \rho \in \mathsf{Env} \setminus \{\bot\}$.

Example 1.2. Consider the program P

$$(\mathtt{x} \geq 0; \mathtt{x++})^*; \mathtt{x} < 0$$

The program never halts on $\forall \rho \in \mathsf{Env}$ s.t. $\rho(\mathtt{x}) \geq 0$. In fact in these cases it builds the transition system in figure 1.1, where the infinite path

$$\langle (\mathtt{x} \geq 0; \mathtt{x++})^*; x < 0, \rho \rangle \to^* \langle (\mathtt{x} \geq 0; \mathtt{x++})^*; x < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + 1] \rangle \to^* \dots$$

$$\dots \to^* \langle (\mathtt{x} \geq 0; \mathtt{x++})^*; x < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + k] \rangle \to^* \dots$$

is always present.

Example 1.3. Consider the program

$$(\mathtt{x++})^*$$

it partially halts ($\langle(\mathtt{x++})^*, \rho\rangle \downarrow$), as according to the transition rule $\mathrm{star}_{\mathrm{fix}}$ $\exists \rho \in \mathsf{Env}$ s.t.

$$\frac{\rho \neq \bot}{\langle(\mathtt{x++})^*, \rho\rangle \to \rho}\ \mathrm{star}_{\mathrm{fix}}$$

But it also partially loops ($\langle(\mathtt{x++})^*, \rho\rangle \uparrow$). In fact we can build the infinite path

$$\langle(\mathtt{x++})^*, \rho[\mathtt{x} \mapsto 0]\rangle \to^* \langle(\mathtt{x++})^*, \rho[\mathtt{x} \mapsto 1]\rangle \to^* \langle(\mathtt{x++})^*, \rho[\mathtt{x} \mapsto 2]\rangle \to^* \dots$$

Other useful lemmas in the system are the composition and decomposition lemma.

Lemma 1.2 (Decomposition lemma)**.** *If $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^k \rho''$, then there exists a state ρ' and a natural number k_1, k_2 s.t. $\langle \mathsf{C}_1, \rho\rangle \to^{k_1} \rho'$ and $\langle \mathsf{C}_2, \rho'\rangle \to^{k_2} \rho''$, where $k_1 + k_2 = k$*

Corollary 1.1. *If $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^* \rho''$ then $\exists \rho'$ s.t. $\langle \mathsf{C}_1, \rho\rangle \to^* \rho'$ and $\langle \mathsf{C}_2, \rho'\rangle \to^* \rho''$.*

Lemma 1.3 (Composition lemma)**.** *If $\langle \mathsf{C}_1, \rho\rangle \to^k \rho'$ then $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^k \langle \mathsf{C}_2, \rho'\rangle$*

Corollary 1.2. *If $\langle \mathsf{C}_1, \rho\rangle \to^* \rho'$ then $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^* \langle \mathsf{C}_2, \rho'\rangle$.*

In order to better talk about the intermediate states in the execution of a program we also introduce the notion of reducts:

Definition 1.9 (Reducts)**.** Let Imp* denotes the set whose elements are statements in Imp. The reduction function $\mathsf{red} : \mathrm{Imp} \to \mathrm{Imp}^*$ is recursively defined by the following clauses:

$$\begin{aligned}
\mathsf{red}(\mathsf{e}) &\triangleq \{\mathsf{e}\} \\
\mathsf{red}(\mathsf{C}_1 + \mathsf{C}_2) &\triangleq \{\mathsf{C}_1 + \mathsf{C}_2\} \cup \mathsf{red}(\mathsf{C}_1) \cup \mathsf{red}(\mathsf{C}_2) \\
\mathsf{red}(\mathsf{C}_1; \mathsf{C}_2) &\triangleq (\mathsf{red}(\mathsf{C}_1); \mathsf{C}_2) \cup \mathsf{red}(\mathsf{C}_2) \\
\mathsf{red}(\mathsf{C}^*) &\triangleq \{\mathsf{C}^*\} \cup (\mathsf{red}(\mathsf{C}); \mathsf{C}^*)
\end{aligned}$$

Where we overload the symbol ; with the operator $; : \mathrm{Imp}^* \times \mathrm{Imp} \to \mathrm{Imp}^*$ defined by

$$\begin{aligned}
\varnothing; \mathsf{C} &\triangleq \varnothing \\
\{\mathsf{C}_1, \dots, \mathsf{C}_k\}; \mathsf{C} &\triangleq \{\mathsf{C}_1; \mathsf{C}, \dots, \mathsf{C}_k; \mathsf{C}\}
\end{aligned}$$

Notice that the set of reduction of any finite program $\mathsf{C} \in \mathrm{Imp}$ is finite.

1.4 Functions in Imp

In the following section we argue that the set of functions is at least a superset of the partially recursive functions described in [Cut80]. This way we can derive some results from well known computability results, without proving them from scratch. We can do this by encoding partial recursive functions into Imp programs. Partial recursive functions are functions $\mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ with airity k:

Definition 1.10 (Partially recursive functions)**.** The class $\mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ of *partially recursive functions* is the least class of functions on the natural numbers which contains

(a) the zero function:

$$\begin{aligned}
z : \mathbb{N}^k &\to \mathbb{N} \\
(x_1, \dots, x_k) &\mapsto 0
\end{aligned}$$

(b) the successor function

$$\begin{aligned} s &: \mathbb{N} \to \mathbb{N} \\ & x_1 \mapsto x_1 + 1 \end{aligned}$$

(c) the projection function

$$\begin{aligned} U_i^k &: \mathbb{N}^k \to \mathbb{N} \\ & (x_1, \dots, x_k) \mapsto x_i \end{aligned}$$

and is closed under

(1) composition: given a function $f : \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ and functions $g_1, \dots, g_k : \mathbb{N}^n \overset{r}{\hookrightarrow} \mathbb{N}$ the *composition* $h : \mathbb{N}^n \overset{r}{\hookrightarrow} \mathbb{N}$ is defined by

$$h(\vec{x}) = \begin{cases} f(g_1(\vec{x}), \dots, g_k(\vec{x})) & \text{if } g_1(\vec{x}) \downarrow, \dots, g_k(\vec{x}) \downarrow \text{ and } f(g_1(\vec{x}), \dots, g_k(\vec{x})) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

(2) primitive recursion: given $f : \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ and $g : \mathbb{N}^{k+2} \overset{r}{\hookrightarrow} \mathbb{N}$ we define $h : \mathbb{N}^{k+1} \overset{r}{\hookrightarrow} \mathbb{N}$ by *primitive recursion* by

$$\begin{cases} h(\vec{x}, 0) & = f(\vec{x}) \\ h(\vec{x}, y + 1) & = g(\vec{x}, y, h(\vec{x}, y)) \end{cases}$$

(3) minimalization: given $f : \mathbb{N}^{k+1} \overset{r}{\hookrightarrow} \mathbb{N}$, $h : \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ defined trough *unbounded minimalization* is

$$h(\vec{x}) = \mu y. f(\vec{x}, y) = \begin{cases} \text{least } z \text{ s.t.} & \begin{cases} f(\vec{x}, z) = 0 \\ f(\vec{x}, z) \downarrow \quad f(\vec{x}, z') \neq 0 \quad \forall z < z' \end{cases} \\ \uparrow & \text{otherwise} \end{cases}$$

We also need to define what it means providing $(a_1, \dots, a_k)$ as input for an Imp program. We do this by special input states and variables: we can consider initial states $\rho = [\mathtt{x}_1 \mapsto a_1, \dots, \mathtt{x}_k \mapsto a_k]$ where each special variable $\mathtt{x}_k$ maps to its initial value a_k, this way we can encode partial functions input into initial states for a program C. Observe that since we are interested in finite programs, it makes sense to consider only finite collections of free variables.

We also need to define what we mean by program output.

Notation 1.2 (Program output)**.** Let $\mathsf{Env} \ni \rho = [\mathtt{x}_1 \mapsto a_1, \dots, \mathtt{x}_n \mapsto a_n]$. We say

$$\begin{aligned} \langle \mathsf{C}, \rho \rangle \downdownarrows b &\iff \forall \rho' \mid \langle \mathsf{C}, \rho \rangle \to^* \rho' \quad \rho'(\mathtt{y}) = b \\ \langle \mathsf{C}, \rho \rangle \downarrow b &\iff \exists \rho' \mid \langle \mathsf{C}, \rho \rangle \to^* \rho' \quad \rho'(\mathtt{y}) = b \end{aligned}$$

C universally (partially) halts on b whenever for every (for some) final state ρ $\rho(\mathtt{y}) = b$. In other words we are using the special variable $\mathtt{y}$ as an output register.

Definition 1.11 (Imp computability)**.** let $f : \mathbb{N}^k \to \mathbb{N}$ be a function. f is Imp computable if

$$\exists \mathsf{C} \in \text{Imp} \mid \forall (a_1, \dots, a_k) \in \mathbb{N}^k \wedge b \in \mathbb{N}$$

$$\langle \mathsf{C}, \rho \rangle \downdownarrows b \iff (a_1, \dots, a_k) \in \mathit{dom}(f) \wedge f(a_1, \dots, a_k) = b$$

with $\rho = [\mathtt{x}_1 \mapsto a_1, \dots, \mathtt{x}_k \mapsto a_k]$.

We argue that the class of function computed by Imp is the same as the set of partially recursive functions $\mathbb{N} \overset{r}{\hookrightarrow} \mathbb{N}$ (as defined in [Cut80]). To do that we have to prove that the class of functions computed by the Imp language is a *rich* , i.e.

Definition 1.12 (Rich class)**.** A class of functions $\mathcal{A}$ is said to be rich if it includes (a),(b) and (c) and it is closed under (1), (2) and (3).

Lemma 1.4 (Imp functions richness)**.** *The class of Imp-computable function is rich.*

Proof. We proceed by proving that Imp has each and every one of the basic functions (zero, successor, projection).

- The zero function:

$$\begin{aligned} z : \mathbb{N}^k &\to \mathbb{N} \\ (x_1, \dots, x_k) &\mapsto 0 \end{aligned}$$

  is Imp-computable:

$$z(a_1, \dots, a_k) \triangleq y := 0$$

- The successor function

$$\begin{aligned} s : \mathbb{N} &\to \mathbb{N} \\ x_1 &\mapsto x_1 + 1 \end{aligned}$$

  is Imp-computable:

$$s(a_1) \triangleq y := x_1 + 1$$

- The projection function

$$\begin{aligned} U_i^k : \mathbb{N}^k &\to \mathbb{N} \\ (x_1, \dots, x_k) &\mapsto x_i \end{aligned}$$

  is Imp-computable:

$$U_i^k(a_1, \dots, a_k) \triangleq y := x_i + 0$$

We then prove that it is closed under composition, primitive recursion and unbounded minimalization.

Lemma 1.5. *let $f : \mathbb{N}^k \to \mathbb{N}$, $g_1, \dots, g_k : \mathbb{N}^n \to \mathbb{N}$ and consider the composition*

$$\begin{aligned} h : \mathbb{N}^k &\to \mathbb{N} \\ \vec{x} &\mapsto f(g_1(\vec{x}), \dots, g_k(\vec{x})) \end{aligned}$$

h is Imp-computable.

Proof. Since by hp $f, g_n \forall n \in [1, k]$ are computable, we consider their programs $F, G_n \forall n \in [1, k]$. Now consider the program

$$\begin{aligned} &G_1(\vec{x}); \\ &y_1 := y + 0; \\ &G_2(\vec{x}); \\ &y_2 := y + 0; \\ &\dots; \\ &G_k(\vec{x}); \\ &y_k := y + 0; \\ &F(y_1, y_2, \dots, y_k); \end{aligned}$$

Which is exactly h. Therefore Imp is closed under generalized composition. □

Lemma 1.6. *Given $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^{k+2} \to \mathbb{N}$ Imp computable, we argue that $h : \mathbb{N}^{k+1} \to \mathbb{N}$*

$$\begin{cases} h(\vec{x}, 0) = f(\vec{x}) \\ h(\vec{x}, y + 1) = g(\vec{x}, y, h(\vec{x}, y)) \end{cases}$$

defined trough primitive recursion is Imp-computable.

Proof. We want a program to compute $h : \mathbb{N}^{k+1} \to \mathbb{N}$. By hypothesis we have programs F, G to compute respectively $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^{k+2} \to \mathbb{N}$. Consider the program $H(\vec{x}, x_{k+1})$:

$$\begin{aligned}
&s := 0;\\
&F(\vec{x});\\
&(x_{k+1} \notin [0,0]; G(\vec{x}, s, y); s := s+1; x_{k+1} := x_{k+1} - 1)^*;\\
&x_{k+1} \in [0,0];
\end{aligned}$$

which computes exactly h. Therefore Imp is closed under primitive recursion. □

Lemma 1.7. *Let $f : \mathbb{N}^{k+1} \to \mathbb{N}$ be a Imp-computable function. Then the function $h : \mathbb{N}^k \to \mathbb{N}$ defined trough unbounded minimalization*

$$h(\vec{x}) = \mu y. f(\vec{x}, y) = \begin{cases} \textit{least } z \textit{ s.t.} & \begin{cases} f(\vec{x}, z) = 0 \\ f(\vec{x}, z) \downarrow \quad f(\vec{x}, z') \neq 0 \quad \forall z < z' \end{cases} \\ \uparrow & \textit{otherwise} \end{cases} \tag{1.1}$$

is Imp-computable.

Proof. Let F be the program for the computable function f with arity $k+1$, $\vec{x} = (x_1, x_2, \ldots, x_k)$. Consider the program $H(\vec{x})$

$$\begin{aligned}
&z := 0;\\
&F(\vec{x}, z);\\
&(y \notin [0,0]; z := z+1; F(\vec{x}, z))^*;\\
&y \in [0,0];\\
&y := z + 0;
\end{aligned}$$

Which outputs the least z s.t. $F(\vec{x}, z) \downarrow 0$, and loops forever otherwise. Imp is therefore closed under bounded minimalization. □

Since has the zero function, the successor function, the projections function and is closed under composition, primitive recursion and unbounded minimalization, the class of Imp-computable functions is rich. □

Since it is rich and $\mathbb{N} \overset{r}{\hookrightarrow} \mathbb{N}$ is the least class of rich functions, $\mathbb{N} \overset{r}{\hookrightarrow} \mathbb{N} \subseteq \text{Imp}_f$ holds. Therefore we can say

$$f \in \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N} \Rightarrow \exists \mathsf{C} \in \text{Imp} \mid \langle \mathsf{C}, \rho \rangle \downdownarrows b \iff f(a_1, \ldots, a_k) \downarrow b$$

with $\rho = [\mathtt{x}_1 \mapsto a_1, \ldots, \mathtt{x}_k \mapsto a_k]$. From this we get a couple of facts that derive from well known computability results:

Corollary 1.3. $\langle \mathsf{C}, \rho \rangle \upuparrows$ *(i.e., $\langle \mathsf{C} \rangle X = \varnothing$) is undecidable.*

Proof. The set of functions $f \in \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ s.t. $f(x) \uparrow \forall x \in \mathbb{N}^k$ is not trivial and saturated, therefore it is not recursive by Rice's theorem [Ric53] □

Corollary 1.4. $\langle \mathsf{C}, \rho \rangle \downdownarrows$ *is undecidable.*

Proof. The set of functions $f \in \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ s.t. $f(x) \downarrow \forall x \in \mathbb{N}^k$ is not trivial and saturated, therefore it is not recursive by Rice's theorem [Ric53]; □

1.5 Deciding invariant finiteness

Lemma 1.8. *If $\mathsf{D} \in Imp_{\neq\star}$, and $X \in 2^{env}$ is finite, then*

(i). $\langle \mathsf{D} \rangle X$ *is finite;*

(ii). $\forall \rho \in X \; \langle \mathsf{D}, \rho \rangle \downdownarrows$

(iii). $|\langle \mathsf{D}, \rho \rangle| < \infty$ *for all $\rho \in X$.*

Proof. By induction on the program D:

Base case:
$\mathsf{D} \equiv e$, therefore

(i). $\langle \mathsf{e} \rangle X = \{ (\!|e|\!)\rho \mid \rho \in X, (\!|e|\!)\rho \neq \bot \}$, which is finite, since X is finite;

(ii). by expr rule $\forall \rho \in X$ either $\langle e, \rho \rangle \to (\!|e|\!)\rho$ or $\langle e, \rho \rangle \not\to$. In both cases there are no infinite paths, and therefore $\langle \mathsf{e}, \rho \rangle \downarrow\!\downarrow$;

(iii). Notice that $\langle \mathsf{e}, \rho \rangle = \{ \tau \in \mathsf{Path}^\infty \mid \tau_0 = \langle \mathsf{e}, \rho \rangle \}$ for all $\rho \in X$, therefore $|\langle \mathsf{e}, \rho \rangle| = |X| < \infty$ because of (i).

Inductive cases:

1. $\mathsf{D} \equiv \mathsf{D}_1 + \mathsf{D}_2$, therefore

   (i). $\langle \mathsf{D}_1 + \mathsf{D}_2 \rangle X = \langle \mathsf{D}_1 \rangle X \cup \langle \mathsf{D}_2 \rangle X$. By inductive hypothesis, both $\langle \mathsf{D}_1 \rangle X, \langle \mathsf{D}_2 \rangle X$ are finite, as they are sub expressions of D. Since the union of finite sets is finite, $\langle \mathsf{D}_1 + \mathsf{D}_2 \rangle X$ is finite;

   (ii). by inductive hypothesis again $\forall \rho \in X$ $\langle \mathsf{D}_1, \rho \rangle \downarrow\!\downarrow$ and $\langle \mathsf{D}_2, \rho \rangle \downarrow\!\downarrow$. By sum$_1$ rule $\langle \mathsf{C}_1 + \mathsf{C}_2, rho \rangle \to \langle \mathsf{C}_1, \rho \rangle$ and by sum$_2$ $\langle \mathsf{C}_1 + \mathsf{C}_2, rho \rangle \to \langle \mathsf{C}_2, \rho \rangle$. Therefore $\langle \mathsf{C}_1 + \mathsf{C}_2, \rho \rangle \downarrow\!\downarrow$.

   (iii). For the latter argument, since both $\langle \mathsf{D}_1, \rho \rangle$ and $\langle \mathsf{D}_2, \rho \rangle$ are finite and composed of finite paths $|\langle (\mathsf{D}_1 + \mathsf{D}_2), \rho \rangle| < \infty$.

2. $\mathsf{D} \equiv \mathsf{D}_1; \mathsf{D}_2$, therefore

   (i). $\langle \mathsf{D}_1; \mathsf{D}_2 \rangle X = \langle \mathsf{D}_2 \rangle (\langle \mathsf{D}_1 \rangle X)$. By inductive hypothesis $\langle \mathsf{D}_1 \rangle X = Y$. By inductive hypothesis again $\langle \mathsf{D}_2 \rangle Y$ is finite;

   (ii). by inductive hypothesis both $\forall \rho \in X$ $\langle \mathsf{D}_1, \rho \rangle \downarrow\!\downarrow$ and $\forall \rho' \in Y$ $\langle \mathsf{D}_2, \rho' \rangle \downarrow\!\downarrow$, therefore by composition lemma $\langle \mathsf{D}_1; \mathsf{D}_2, \rho \rangle \downarrow\!\downarrow$

   (iii). by inductive hypothesis both $|\langle \mathsf{C}_1, \rho \rangle| < \infty$ and $|\langle \mathsf{C}_2, \rho' \rangle| < \infty$ $\forall \rho \in X, \rho' \in \langle \mathsf{C}_1 \rangle X$, todo

□

Lemma 1.9. *Given $\mathsf{D} \in Imp_{\neq \star}$, and $X \in 2^{env}$, the predicate "$\langle \mathsf{D}^* \rangle X$ is finite" is undecidable.*

Proof. Suppose we can decide $\langle \mathsf{D}^* \rangle X$ is finite. We show that we in that case we would be also able to decide whether $\langle \mathsf{D}^*, \rho \rangle \downarrow\!\downarrow$ for some $\rho \in X$, which is undecidable.

- In case $\langle \mathsf{D}^* \rangle X$ is infinite, then it must be that $\forall k \in \mathbb{N}$

  $$\langle \mathsf{D} \rangle^{k+1} X \not\subseteq \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^i X$$

  otherwise if for some k $\langle \mathsf{D} \rangle^{k+1} X \subseteq \cup_{i=0}^{k} \langle \mathsf{D} \rangle^i X$ it would mean that todo

  we would reach a fixpoint and $\langle \mathsf{D}^* \rangle X$ would be finite. Since each application of D must create an nonempty set of new environments, we can build the inductive sequence

  $$Y_0 = X$$
  $$Y_{k+1} = (\langle \mathsf{D} \rangle Y_k) \setminus Y_k$$

  where $\forall \rho' \in Y_{k+1} \exists \rho \in Y_k \mid \rho' \in \langle \mathsf{D} \rangle \rho$ by definition. This means that there must be at least one $\rho_1 \in X$ that produces an infinite path

  $$\langle \mathsf{D}^*, \rho_1 \rangle \to^* \langle \mathsf{D}^*, \rho_2 \rangle \to^* \dots$$

  which produces new environments at each application of D: $\rho_1, \rho_2, \dots \mid \forall i, j \in \mathbb{N}$ $\rho_i \neq \rho_j$ and therefore $\langle \mathsf{D}^*, \rho_1 \rangle \uparrow$ which means that $\langle \mathsf{D}^*, \rho_1 \rangle \downarrow\!\downarrow$ is false.

- In case $\langle \mathsf{D}^* \rangle X$ is finite we can search for infinite paths in $\langle \mathsf{D}^*, \rho \rangle$. Since $\langle \mathsf{D}^* \rangle X = \bigcup_{\rho \in X} \langle \mathsf{D}^* \rangle \{\rho\}$, for every $\rho' \in X$, $\langle \mathsf{D} \rangle \{\rho\}$ is finite. Consider the states in $\langle \mathsf{D} \rangle \{\rho\}$. For each one of them either
  - $\langle \mathsf{D}, \rho' \rangle \rightarrow^* \langle \mathsf{D}', \rho'' \rangle \not\rightarrow$, for every $\rightarrow$ step we can apply the comp$_1$ rule and therefore we would build
    $$\langle \mathsf{D}; \mathsf{D}^*, \rho' \rangle \rightarrow^* \langle \mathsf{D}'; \mathsf{D}^*, \rho'' \rangle \not\rightarrow$$
    which is a finite path, therefore is not interesting.
  - or $\langle \mathsf{D}, \rho' \rangle \rightarrow^* \rho''$ then by composition lemma $\langle \mathsf{D}; \mathsf{D}^*, \rho' \rangle \rightarrow^* \langle \mathsf{D}^*, \rho'' \rangle$ and by star$_{\text{fix}}$ $\langle \mathsf{D}^*, \rho'' \rangle \rightarrow \rho''$ and so $\rho'' \in \langle \mathsf{D}^* \rangle \{\rho\}$.

□

Chapter 2

Intervals

2.1 Interval Analysis

We define *interval analysis* of the above language Imp in a standard way, taking the best correct approximations (bca) for the basic expressions in Exp.

Definition 2.1 (Integer intervals)**.** We call

$$\mathit{Int} \triangleq \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\} \wedge b \in \mathbb{Z} \cup \{+\infty\} \wedge a \leqslant b\} \cup \{\bot^\sharp\}$$

set of integer intervals.

Definition 2.2 (Concretization map)**.** We define the *concretization map* $\gamma : \mathit{Int} \to 2^{\mathbb{Z}}$ as

$$\begin{aligned} \gamma([a,b]) &\triangleq \{x \in \mathbb{Z} \mid a \leqslant x \leqslant b\} \\ \gamma(\bot) &\triangleq \varnothing \end{aligned}$$

Observe that $\langle \mathit{Int}, \sqsubseteq \rangle$ is a complete lattice where for all $I, J \in \mathit{Int}$, $I \sqsubseteq J$ iff $\gamma(I) \subseteq \gamma(J)$.

Definition 2.3 (Abstract integer domain)**.** Let $\mathit{Int}_* \triangleq \mathit{Int} \setminus \{\bot^\sharp\}$. The abstract domain $\mathbb{A}$ for program analysis is the variable-wise lifting of Int:

$$\mathbb{A} \triangleq (\mathit{Var} \to \mathit{Int}_*) \cup \{\bot^\sharp\}$$

where the intervals for a given variable are always nonempty, while $\bot^\sharp$ represents the empty set of environments. Thus, the corresponding concretization is defined as follows:

Definition 2.4 (Interval concretization)**.** We define the *concretization map* for the abstract domain $\mathbb{A}$ $\gamma_{\mathit{Int}} : \mathbb{A} \to 2^{\mathsf{Env}}$ as

$$\begin{aligned} \gamma_{\mathit{Int}}(\bot) &\triangleq \varnothing \\ \forall \eta \neq \bot \quad \gamma_{\mathit{Int}}(\eta) &\triangleq \{\rho \in \mathsf{Env} \mid \forall x \in \mathit{Var}\ \rho(x) \in \gamma(\eta(x))\} \end{aligned}$$

Observation 2.1. If we consider the ordering $\sqsubseteq$ on $\mathbb{A}$ s.t.

$$\forall \eta, \vartheta \in \mathbb{A} \quad \eta \sqsubseteq \vartheta \iff \gamma_{\mathit{Int}}(\eta) \subseteq \gamma_{\mathit{Int}}(\vartheta)$$

then $\langle \mathbb{A}, \sqsubseteq \rangle$ is a complete lattice.

Definition 2.5 (Interval abstraction)**.** We define the *abstraction map* of some numerical set $X \subseteq \mathbb{Z}$ into the abstract domain $\mathbb{A}$: $\alpha_{\mathit{Int}} : 2^{\mathbb{Z}} \to \mathbb{A}$ as

$$\alpha_{\mathit{Int}}(X) \triangleq \begin{cases} \bot^\sharp & \text{if } X = \varnothing \\ [\min X, \max X] & \text{otherwise} \end{cases}$$

Observe that since we have both a concretization map γ_{Int} and an abstraction map α_{Int} we have built the Galois Connection

$$\langle \gamma_{Int}, \mathbb{C}, \mathbb{A}, \alpha_{Int} \rangle$$

between the concrete domain $\mathbb{C}$ and the abstract domain $\mathbb{A}$, resulting

Definition 2.6 (Abstract operations)**.** We define sound abstract operations in the $\mathbb{A}$ domain:

$$\begin{aligned} [a,b] \cup^\sharp [c,d] &\triangleq [\min(a,c), \max(b,d)] \\ [a,b] \cap^\sharp [c,d] &\triangleq \begin{cases} [\max(a,c), \min(b,d)] & \text{if } \min < \max \\ \bot^\sharp & \text{otherwise} \end{cases} \end{aligned}$$

And sound abstract arithmetical operations:

$$\begin{aligned} -^\sharp\ [a,b] &\triangleq [-b,-a] \\ [a,b]+^\sharp[c,d] &\triangleq [a+c, b+d] \\ [a,b]-^\sharp[c,d] &\triangleq [a-c, b-d] \\ [a,b]\times^\sharp[c,d] &\triangleq [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)] \end{aligned}$$

Definition 2.7 (Interval sharpening)**.** For a nonempty interval $[a,b] \in Int$ and $c \in \mathbb{Z}$, we define two operations raising $\uparrow$ the lower bound to c and lowering $\downarrow$ the upper bound to c, respectively:

$$\begin{aligned} [a,b] \uparrow c &\triangleq \begin{cases} [\max\{a,c\}, b] & \text{if } c \leqslant b \\ \bot & \text{if } c > b \end{cases} \\ [a,b] \downarrow c &\triangleq \begin{cases} [a, \min\{b,c\}] & \text{if } c \geq a \\ \bot & \text{if } c < a \end{cases} \end{aligned}$$

Observe that $\max([a,b] \downarrow c) \leqslant c$ always holds. □

Definition 2.8 (Interval addition and subtraction)**.** For a nonempty interval $[a,b] \in Int$ and $c \in \mathbb{Z}$ define $[a,b] \pm c \triangleq [a \pm c, b \pm c]$ (recall that $\pm\infty + c = \pm\infty - c = \pm\infty$). □

Observe that for every interval $[a,b] \in Int$ and $c \in \mathbb{Z}$

$$\max([a,b] \uparrow c) \leqslant b \qquad \text{and} \qquad \max([a,b] \downarrow c) \leqslant c$$

that trivially holds by defining $\max(\bot) \triangleq 0$ (i.e., 0 is the maximum of an empty interval).

The *interval semantics* of Imp is defined as the strict (i.e., preserving $\bot$) extension of the following function $[\![\cdot]\!] : \exp \cup \mathrm{Imp} \to \mathbb{A} \to \mathbb{A}$. For all $\eta : Var \to Int_*$,

$$
\begin{aligned}
[\![\mathtt{x} \in S]\!]\eta &\triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \sqcap \alpha_{Int}(S)] & \text{if } \eta(\mathtt{x}) \sqcap \alpha_{Int}(S) \neq \bot \\ \bot & \text{otherwise} \end{cases} \\
[\![\mathtt{x} \in [a,b]]\!]\eta &\triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \sqcap [a,b]] & \text{if } \eta(\mathtt{x}) \sqcap [a,b] \neq \bot \\ \bot & \text{otherwise} \end{cases} \\
[\![\mathtt{x} \leqslant k]\!]\eta &\triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \downarrow k] & \text{if } \eta(\mathtt{x}) \downarrow k \neq \bot \\ \bot & \text{otherwise} \end{cases} \\
[\![\mathtt{x} > k]\!]\eta &\triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \uparrow k+1] & \text{if } \eta(\mathtt{x}) \downarrow k \neq \bot \\ \bot & \text{otherwise} \end{cases} \\
[\![\mathsf{true}]\!]\eta &\triangleq \eta \\
[\![\mathsf{false}]\!]\eta &\triangleq \bot \\
[\![\mathtt{x} := k]\!]\eta &\triangleq \eta[\mathtt{x} \mapsto [k,k]] \\
[\![\mathtt{x} := \mathtt{y} + k]\!]\eta &\triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) + k] \\
[\![\mathtt{x} := \mathtt{x} - k]\!]\eta &\triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) - k] \\
[\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta &\triangleq [\![\mathsf{C}_1]\!]\eta \sqcup [\![\mathsf{C}_2]\!]\eta \\
[\![\mathsf{C}_1 ; \mathsf{C}_2]\!]\eta &\triangleq [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\eta) \\
[\![\mathsf{C}^*]\!]\eta &\triangleq \textstyle\bigsqcup_{i \in \mathbb{N}} [\![\mathsf{C}]\!]^i(\eta) \\
[\![\mathsf{fix}(\mathsf{C})]\!]\eta &\triangleq \mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu)
\end{aligned}
$$

The semantics is well-defined, because of the following lemma:

Lemma 2.1. *for all* $\mathsf{C} \in Imp$,

$$[\![\mathsf{C}]\!] : \mathbb{A} \to \mathbb{A}$$

is monotone.

Proof. What we have to proof is that given $\eta, \vartheta \in \mathbb{A}$, with $\eta \sqsubseteq \vartheta$ then $\forall \mathsf{C} \in \mathrm{Imp}\ [\![\mathsf{C}]\!]\eta \sqsubseteq [\![\mathsf{C}]\!]\vartheta$. We will work by induction on the grammar of C:

Base cases:

We avoid cases where $\eta = \bot$ and $[\![\mathsf{C}]\!]\eta = \bot$ as $\forall \vartheta \in \mathbb{A}\ \bot \sqsubseteq \vartheta$ and it becomes trivially true.

- $\mathsf{C} \equiv \mathtt{x} \in S$. Then

$$
\begin{aligned}
[\![\mathtt{x} \in S]\!]\eta &= \eta[x \mapsto \eta(x) \sqcap Int(S)] \\
[\![\mathtt{x} \in S]\!]\vartheta &= \vartheta[x \mapsto \vartheta(x) \sqcap Int(S)]
\end{aligned}
$$

Since $\eta(x) \sqcap Int(S) \neq \bot$ and $\eta \sqsubseteq \vartheta$, then $\vartheta(x) \sqcap Int(S) \neq \bot$. We can see that

$$
\begin{aligned}
\eta \sqsubseteq \vartheta &\iff \gamma(\eta) \subseteq \gamma(\vartheta) \\
&\iff \{x \in \mathbb{Z} \mid x \in \eta(\mathtt{x})\} \subseteq \{x \in \mathbb{Z} \mid x \in \vartheta(\mathtt{x})\} \\
&\iff \{x \in \mathbb{Z} \mid x \in \eta(\mathtt{x})\} \cap \{x \in \mathbb{Z} \mid x \in Int(S)\} \subseteq \{x \in \mathbb{Z} \mid x \in \vartheta(\mathtt{x})\} \cap \{x \in \mathbb{Z} \mid x \in Int(S)\} \\
&\iff \{x \in \mathbb{Z} \mid x \in \eta(\mathtt{x}) \wedge x \in Int(S)\} \subseteq \{x \in \mathbb{Z} \mid x \in \vartheta(\mathtt{x}) \wedge x \in Int(S)\} \\
&\iff \{x \in \mathbb{Z} \mid x \in \eta(\mathtt{x}) \sqcap Int(S)\} \subseteq \{x \in \mathbb{Z} \mid x \in \vartheta(\mathtt{x}) \sqcap Int(S)\} \\
&\iff \gamma_{Int}(\eta[x \mapsto \eta(\mathtt{x}) \sqcap Int(S)](\mathtt{x})) \subseteq \gamma_{Int}(\vartheta[x \mapsto \vartheta(\mathtt{x}) \sqcap Int(S)](\mathtt{x})) \\
&\iff [\![\mathtt{x} \in S]\!]\eta \sqsubseteq [\![\mathtt{x} \in S]\!]\vartheta
\end{aligned}
$$

- for the base cases $\mathtt{x} \in [a,b], \mathtt{x} \leqslant k, \mathtt{x} > k$ we can use the same proceedings;
- $\mathsf{C} \equiv \mathsf{true}$. Then $[\![\mathsf{true}]\!]\eta = \eta \sqsubseteq \vartheta = [\![\mathsf{true}]\!]\vartheta$;

- $\mathsf{C} \equiv \mathsf{false}$. Then $\llbracket \mathsf{false} \rrbracket \eta = \bot \sqsubseteq \bot = \llbracket \mathsf{false} \rrbracket \vartheta$;

- $\mathsf{C} \equiv \mathtt{x} := k$. Then

$$\begin{aligned} \eta \sqsubseteq \vartheta \iff & \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\vartheta) \\ \iff & \{\rho \in \mathsf{Env} \mid \forall \mathtt{x} \in \mathit{Var} \rho(\mathtt{x}) \in \gamma(\eta(\mathtt{x}))\} \subseteq \{\rho \in \mathsf{Env} \mid \forall \mathtt{x} \in \mathit{Var} \rho(\mathtt{x}) \in \gamma(\vartheta(\mathtt{x}))\} \\ \iff & \forall \mathtt{x} \in \mathit{Var}, \rho \in \mathsf{Env} \quad \rho(\mathtt{x}) \in \gamma(\eta(\mathtt{x})) \Rightarrow \rho(\mathtt{x}) \in \gamma(\vartheta(\mathtt{x})) \end{aligned} \tag{2.1}$$

Notice that

$$\llbracket \mathtt{x} := k \rrbracket \eta = \eta[\mathtt{x} \mapsto [k, k]]$$

$$\llbracket \mathtt{x} := k \rrbracket \vartheta = \vartheta[\mathtt{x} \mapsto [k, k]]$$

because of equation 2.1 in this case we know that $\forall \mathtt{y} \in \mathit{Var},\ \mathtt{y} \neq \mathtt{x}\ \rho(\mathtt{y}) \in \gamma(\eta(\mathtt{y})) \Rightarrow \rho(\mathtt{y}) \in \gamma(\vartheta(\mathtt{y}))$. For $\mathtt{x}$ it holds that $\rho(\mathtt{x}) \in \gamma([k, k]) \Rightarrow \rho(\mathtt{x}) \in \gamma([k, k])$ and therefore

$$\begin{aligned} & \forall \mathtt{y} \in \mathit{Var}, \rho \in \mathsf{Env} \quad \rho(\mathtt{y}) \in \gamma(\eta[\mathtt{x} \mapsto [k, k]](\mathtt{y})) \Rightarrow \rho(\mathtt{y}) \in \gamma(\vartheta[\mathtt{x} \mapsto [k, k]](\mathtt{y})) \\ & \iff \gamma_{Int}(\llbracket \mathtt{x} := k \rrbracket \eta) \subseteq \gamma_{Int}(\llbracket \mathtt{x} := k \rrbracket \vartheta) \\ & \iff \llbracket \mathtt{x} := k \rrbracket \eta \sqsubseteq \llbracket \mathtt{x} := k \rrbracket \vartheta \end{aligned}$$

- For $\mathsf{C} \equiv \mathtt{x} := \mathtt{y} + k, \mathtt{x} := \mathtt{y} - k$ the procedure is the same.

Recursive cases:

- $\mathsf{C} \equiv \mathsf{C}_1 + \mathsf{C}_2$. Then

$$\begin{aligned} \llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta &= \llbracket \mathsf{C}_1 \rrbracket \eta \sqcup \llbracket \mathsf{C}_2 \rrbracket \eta \\ &\sqsubseteq \llbracket \mathsf{C}_1 \rrbracket \vartheta \sqcup \llbracket \mathsf{C}_2 \rrbracket \vartheta && \text{by inductive hp.} \\ &= \llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \vartheta \end{aligned}$$

- $\mathsf{C} \equiv \mathsf{C}_1; \mathsf{C}_2$. Then

$$\begin{aligned} \llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta = {} & \llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta) \\ & \alpha = \llbracket \mathsf{C}_1 \rrbracket \eta \sqsubseteq \llbracket \mathsf{C}_1 \rrbracket \vartheta = \beta && \text{by inductive hp.} \\ & \llbracket \mathsf{C}_2 \rrbracket \alpha \sqsubseteq \llbracket \mathsf{C}_2 \rrbracket \beta && \text{by inductive hp.} \\ & \llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta) \sqsubseteq \llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \vartheta) && \text{by substitution} \end{aligned}$$

- C^*. Then by inductive hypothesis $\forall i \in \mathbb{N}. \llbracket \mathsf{C} \rrbracket^i \eta \sqsubseteq \llbracket \mathsf{C} \rrbracket^i \vartheta$, which means

$$\llbracket \mathsf{C}^* \rrbracket \eta = \bigsqcup_{i \in \mathbb{N}} \llbracket \mathsf{C} \rrbracket^i \eta \sqsubseteq \bigsqcup_{i \in \mathbb{N}} \llbracket \mathsf{C} \rrbracket^i \vartheta = \llbracket \mathsf{C}^* \rrbracket \vartheta.$$

□

Example 2.1. This is the case, for instance, the following program P represents the difference between the Kleene Star and the Fix operator:

```
while x < 8 do
if x = 2 then x := x+6;
x := x-3
if x <= 0 then x:=0
```

starting with the finite interval $[3,4]$ we get the following loop invariants:

$$\begin{aligned} \text{Kleene: } & \sqcup\{[3,4],[0,1],[0,0],[0,0],\ldots\} = [0,4] \\ \text{Fix: } & \sqcup\{\bot,[3,4],[0,4],[0,5],[0,5],\ldots\} = [0,5] \end{aligned}$$

Both invariants are correct, because they over-approximate the most precise concrete invariant $\{0,1,3,4\}$, however the Kleene invariant is strictly more precise than the Fix one.

Lemma 2.2 ($\mathsf{fix(C)}$ **is syntactic sugar**)**.** *For all* η, $[\![\mathsf{fix(C)}]\!]\eta = [\![(\mathsf{true} + \mathsf{C})^*]\!]\eta$.

Proof. Let us first show by induction that

$$\forall i \geq 0.\ (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot = (\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\eta \tag{$\sharp$}$$

$i = 0$: $(\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^1\bot = \eta \sqcup \bot \sqcup [\![\mathsf{C}]\!]\bot = \eta = (\mathsf{true} \sqcup [\![\mathsf{C}]\!])^0\eta$.
$i+1$:

$$\begin{aligned} (\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\eta &= \\ (\mathsf{true} \sqcup [\![\mathsf{C}]\!])((\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\eta) &= \\ ((\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\eta) \sqcup [\![\mathsf{C}]\!]((\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\eta) &= && \text{By induction} \\ (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot \sqcup [\![\mathsf{C}]\!]((\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot) &= && \text{As } \eta \sqsubseteq (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot \\ \eta \sqcup (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot \sqcup [\![\mathsf{C}]\!]((\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot) &= \\ (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])((\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot) &= \\ (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+2}\bot \end{aligned}$$

Let us also show that:

$$\mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu) = \mathrm{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup [\![\mathsf{C}]\!]\mu) \tag{$\diamond$}$$

Observe that $\mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu) = \eta \sqcup [\![\mathsf{C}]\!](\mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$, so that we have that:

$$\eta \sqcup \mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu) \sqcup [\![\mathsf{C}]\!](\mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu)) \sqsubseteq \mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu)$$

As a consequence, $\mathrm{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup [\![\mathsf{C}]\!]\mu) \sqsubseteq \mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu)$ holds. The reverse inequality follows because, for all μ, $\eta \sqcup [\![\mathsf{C}]\!]\mu \sqsubseteq \eta \sqcup \mu \sqcup [\![\mathsf{C}]\!]\mu$.

Then, we have that:

$$\begin{aligned} [\![\mathsf{fix(C)}]\!]\eta &= \\ \mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu) &= && \text{By } (\diamond) \\ \mathrm{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup [\![\mathsf{C}]\!]\mu) &= && \text{By Knaster-Tarski Theorem} \\ \bigsqcup_{i\in\mathbb{N}} (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\bot &= \\ \bot \sqcup \bigsqcup_{i\in\mathbb{N}} (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot &= && \text{By (2.3)} \\ \bigsqcup_{i\in\mathbb{N}} (\mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i}\eta &= \\ [\![(\mathsf{true} + \mathsf{C})^*]\!]\eta. \end{aligned}$$

□

Theorem 2.1 (**Correctness**)**.** *For all* $\mathsf{C} \in$ *Imp and* $\eta \in \mathbb{A}$, $\langle\mathsf{C}\rangle\gamma(\eta) \subseteq \gamma([\![\mathsf{C}]\!]\eta)$ *holds.*

Proof. by induction on $\mathsf{C} \in \mathrm{Imp}$:

Base cases:

- $\mathsf{C} \equiv \mathtt{x} \in S$:
  - $\langle \mathtt{x} \in \mathsf{S} \rangle \gamma_{Int}(\eta) = \{\rho \in \mathsf{Env} \mid \forall y \in \mathit{Var}\ \rho(y) \in \gamma(\eta(y))\} \cap \{\rho \in \mathsf{Env} \mid \rho(\mathtt{x}) \in S\}$
  - $\gamma_{Int}(\llbracket \mathtt{x} \in S \rrbracket \eta) = \{\rho \in \mathsf{Env} \mid \forall y \in \mathit{Var}\ \rho(y) \in \gamma(\eta(y))\} \cap \{\rho \in \mathsf{Env} \mid \rho(\mathtt{x}) \in \mathit{Int}(S)\}$

  S is just decidable, not directly an interval, therefore in general $S \subseteq \mathit{Int}(S)$, and therefore

  $$\langle \mathtt{x} \in \mathsf{S} \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathtt{x} \in S \rrbracket \eta);$$

- $\mathsf{C} \equiv \mathtt{x} \in [a, b], \mathtt{x} \leqslant k, \mathtt{x} > k$: is the same as the latter case;
- $\mathsf{C} \equiv \mathsf{true}$: $\langle \mathsf{true} \rangle \gamma_{Int}(\eta) = \gamma_{Int}(\eta)$, $\gamma_{Int}(\llbracket \mathsf{true} \rrbracket \eta) = \gamma_{Int}(\eta)$, and since $\gamma_{Int}(\eta) \subseteq \gamma_{Int}(\eta)$

  $$\langle \mathsf{true} \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathsf{true} \rrbracket \eta);$$

- $\mathsf{C} \equiv \mathsf{false}$: $\langle \mathsf{false} \rangle \gamma_{Int}(\eta) = \varnothing$, $\gamma_{Int}(\llbracket \mathsf{false} \rrbracket \eta) = \varnothing$ and therefore

  $$\langle \mathsf{false} \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathsf{false} \rrbracket \eta);$$

- $\mathsf{C} \equiv \mathtt{x} := k$ therefore $\langle \mathtt{x} := \mathsf{k} \rangle \gamma_{Int}(\eta) = \{\rho \in \mathsf{Env} \mid \forall y \in \mathit{Var}.y \neq \mathtt{x} \Rightarrow \rho(y) \in \gamma(\eta(y)), \rho(\mathtt{x}) \in \gamma(\eta(\mathtt{x}) + k)\} = \gamma_{Int}(\llbracket \mathtt{x} := k \rrbracket \eta)$ therefore

  $$\langle \mathtt{x} := \mathsf{k} \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathtt{x} := k \rrbracket \eta);$$

- $\mathsf{C} \equiv \mathtt{x} := \mathtt{y} + k, \mathtt{x} := \mathtt{y} - k$ is the same as the latter case.

Inductive cases:

- $\mathsf{C} \equiv \mathsf{C}_1 + \mathsf{C}_2$, therefore

  $$\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle \gamma_{Int}(\eta) = \langle \mathsf{C}_1 \rangle \gamma_{Int}(\eta) \cup \langle \mathsf{C}_2 \rangle \gamma_{Int}(\eta)$$

  and

  $$\gamma_{Int}(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta) = \gamma_{Int}(\llbracket \mathsf{C}_1 \rrbracket \eta \sqcup \llbracket \mathsf{C}_2 \rrbracket \eta) = \gamma_{Int}(\llbracket \mathsf{C}_1 \rrbracket \eta) \cup \gamma_{Int}(\llbracket \mathsf{C}_2 \rrbracket \eta).$$

  By inductive hypothesis both $\langle \mathsf{C}_1 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathsf{C}_1 \rrbracket \eta)$ and $\langle \mathsf{C}_2 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathsf{C}_2 \rrbracket \eta)$, therefore

  $$\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta);$$

- $\mathsf{C} \equiv \mathsf{C}_1; \mathsf{C}_2$, therefore $\langle \mathsf{C}_1; \mathsf{C}_2 \rangle \gamma_{Int}(\eta) = \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle \gamma_{Int}(\eta))$, while

  $$\gamma_{Int}(\llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta) = \gamma_{Int}(\llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta)).$$

- $\mathsf{C} \equiv \mathsf{C}^*$, therefore $\langle \mathsf{C}^* \rangle \gamma_{Int}(\eta) = \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i \gamma_{Int}(\eta)$, while $\gamma_{Int}(\llbracket \mathsf{C}^* \rrbracket \eta) = \gamma_{Int}(\bigsqcup_{i \in \mathbb{N}} \llbracket \mathsf{C}^* \rrbracket \eta)$

□

Remark 2.1. Let us remark that in case we were interested in studying termination of the abstract interpreter, we could assume that the input of a program will always be a finite interval in such a way that non termination can be identified with the impossibility of converging to a finite interval for some variable. In fact, starting from an environment η which maps each variable to a finite interval, $\llbracket \mathsf{C} \rrbracket \eta$ might be infinite on some variable when C includes a either Kleene or fix iteration which does not converge in finitely many steps.

2.2 Computing the interval semantics

In this section we argue that for the language Imp the interval abstract semantics is computable in finite time without widening.

Observe that the exact computation provides, already for our simple language, a precision which is not obtainable with (basic) widening and narrowing. In the example below the semantics maps x and y to $[0, 2]$ and $[6, 8]$ resp., while widening/narrowing to $[0, \infty]$ and $[6, \infty]$

```
x:=0;
y:=0;
while (x<=5) do
   if (y=0) then
      y=y+1;
   endif;
   if (x==0) then
      x:=y+7;
   endif;
done;
end
```

Of course, for the collecting semantics this is not the case. Already computing a finite upper bound for loop invariants when they are finite is impossible as this would allow to decide termination, as we have seen in section 1.5.

Problem 2.1 (Termination of interval analysis)**.** Given $\mathsf{C} \in \text{Imp}$, $\eta \in \mathbb{A}$, decide: $[\![\mathsf{C}]\!]\eta =^{?} \top$

First, given a program, we associate each variable with a *single bound*, which captures both both an *upper bound*, for which the rough idea is that, whenever a variable is beyond that bound, the behavior of the program with respect to that variable becomes stable and an *increment bound* which captures the largest increment or decrement that can affect a variable.

Definition 2.9 (**Program bound**)**.** The *bound* associated with a command $\mathsf{C} \in \text{Imp}$ is a natural number, denoted $(\mathsf{C})^{\mathsf{b}} \in \mathbb{N}$, defined inductively as follows:

$$
\begin{aligned}
(\mathtt{x} \in S)^{\mathsf{b}} &\triangleq \begin{cases} \min(S) & \text{if } \max(S) = \infty \\ \max(S) & \text{if } \max(S) \in \mathbb{N} \end{cases} \\
(\mathtt{x} \in [a, b])^{\mathsf{b}} &\triangleq \begin{cases} a & \text{if } b = \infty \\ b & \text{if } b \in \mathbb{N} \end{cases} \\
(\mathtt{x} \leqslant k)^{\mathsf{b}} &\triangleq k \\
(\mathtt{x} > k)^{\mathsf{b}} &\triangleq k \\
(\mathsf{true})^{\mathsf{b}} &\triangleq 0 \\
(\mathsf{false})^{\mathsf{b}} &\triangleq 0 \\
(\mathtt{x} := k)^{\mathsf{b}} &\triangleq k \\
(\mathtt{x} := \mathtt{y} + k)^{\mathsf{b}} &\triangleq k \\
(\mathtt{x} := \mathtt{y} - k)^{\mathsf{b}} &\triangleq k \\
(\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}} &\triangleq (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}} \\
(\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} &\triangleq (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}} \\
(\mathsf{C}^*)^{\mathsf{b}} &\triangleq (|\mathit{vars}(\mathsf{C})| + 1)(\mathsf{C})^{\mathsf{b}}
\end{aligned}
$$

where $\mathit{vars}(\mathsf{C})$ denotes the set of variables occurring in C.

Definition 2.10 (Bound Environment). A bound environment (benv for short) is a total function $b : \mathit{Var} \to \mathbb{N}$. We define $\mathsf{bEnv} \triangleq \{b \mid b : \mathit{Var} \to \mathbb{N}\}$. Each command $\mathsf{C} \in$ Imp induces a benv transformer $[C]^{\mathsf{b}} : \mathsf{bEnv} \to \mathsf{bEnv}$, which is defined inductively as follows:

$$
\begin{aligned}
[\mathtt{x} \in S]^{\mathsf{b}} b &\triangleq \begin{cases} b[\mathtt{x} \mapsto b(\mathtt{x}) + \min(S)] & \text{if } \max(S) = \infty \\ b[\mathtt{x} \mapsto b(\mathtt{x}) + \max(S)] & \text{if } \max(S) \in \mathbb{N} \end{cases} \\
[\mathtt{x} := k]^{\mathsf{b}} b &\triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + k] \\
[\mathtt{x} := \mathtt{y} + k]^{\mathsf{b}} b &\triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + b(\mathtt{y}) + k] \\
[\mathtt{x} := \mathtt{y} - k]^{\mathsf{b}} b &\triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + b(\mathtt{y}) - k] \\
[\mathsf{C}_1 + \mathsf{C}_2]^{\mathsf{b}} b &\triangleq \lambda \mathtt{x}.([\mathsf{C}_1]^{\mathsf{b}} b)(\mathtt{x}) + ([\mathsf{C}_2]^{\mathsf{b}} b)(\mathtt{x}) \\
[\mathsf{C}_1 ; \mathsf{C}_2]^{\mathsf{b}} b &\triangleq \lambda \mathtt{x}.([\mathsf{C}_1]^{\mathsf{b}} b)(\mathtt{x}) + ([\mathsf{C}_2]^{\mathsf{b}} b)(\mathtt{x}) \\
[\mathsf{C}^*]^{\mathsf{b}} b &\triangleq \lambda \mathtt{x}.(|\mathit{vars}(\mathsf{C})| + 1)([\mathsf{C}]^{\mathsf{b}} b)(\mathtt{x})
\end{aligned}
$$

where $\mathit{vars}(\mathsf{C})$ denotes the set of variables occurring in C.

Lemma 2.3. *For all* $\mathsf{C} \in \mathit{Imp}$, $(\mathsf{C})^{\mathsf{b}} = \sum_{\mathtt{x} \in \mathit{vars}(\mathsf{C})} ([\mathsf{C}]^{\mathsf{b}} b_0)(\mathtt{x})$, *with* $b_0 \triangleq \lambda x.0$.

Proof. By induction on $\mathsf{C} \in$ Imp.

Base cases:

$(\mathtt{x} \in S)$:

$$
\begin{aligned}
(\mathtt{x} \in S)^{\mathsf{b}} &= \begin{cases} \min(S) & \text{if } \max(S) = \infty \\ \max(S) & \text{otherwise} \end{cases} \\
[\mathtt{x} \in S]^{\mathsf{b}} b_0 &= \begin{cases} b_0[\mathtt{x} \mapsto 0 + \min(S)] & \text{if } \max(S) = \infty \\ b_0[\mathtt{x} \mapsto 0 + \max(S)] & \text{if } \max(S) \in \mathbb{N} \end{cases}
\end{aligned}
$$

and since $\mathtt{x}$ is the only variable in $\mathit{vars}(\mathtt{x} \in S)$, $(\mathtt{x} \in S)^{\mathsf{b}} = [\mathtt{x} \in S]^{\mathsf{b}} b_0(\mathtt{x})$

$(\mathtt{x} \in [a, b]), (\mathtt{x} \leqslant k), (\mathtt{x} > k)$ are the same as the latter case;

$(\mathsf{true}), (\mathsf{false})$: notice that $\mathit{vars}(\mathsf{true}) = \mathit{vars}(\mathsf{false}) = \varnothing$;

$(\mathtt{x} := k)$: just notice that $(\mathtt{x} := k)^{\mathsf{b}} = k = b_0(\mathtt{x}) + k = b_0[\mathtt{x} \mapsto b_0 + k] = [\mathtt{x} := k]^{\mathsf{b}} b_0$ and $\mathtt{x}$ is the only variable in $\mathtt{x} := k$.

$(\mathtt{x} := \mathtt{x} + k), (\mathtt{x} := \mathtt{x} - k)$ are analogous to the latter case.

Inductive cases:

$(\mathsf{C}_1 + \mathsf{C}_2)$

$$\begin{aligned}
(\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}} &= \\
(\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}} &= && \text{by inductive hypothesis} \\
\sum_{\mathtt{x} \in vars(\mathsf{C}_1)} ([\mathsf{C}]^{\mathsf{b}} b_0)(\mathtt{x}) + \sum_{\mathtt{x} \in vars(\mathsf{C}_2)} ([\mathsf{C}]^{\mathsf{b}} b_0)(\mathtt{x}) &= \\
\sum_{\mathtt{x} \in vars(\mathsf{C}_1) \cap vars(\mathsf{C}_2)} ([\mathsf{C}_1]^{\mathsf{b}} b_0)(\mathtt{x}) + ([\mathsf{C}_2]^{\mathsf{b}} b_0)(\mathtt{x}) &+ \\
\sum_{\mathtt{x} \in vars(\mathsf{C}_1) \smallsetminus vars(\mathsf{C}_2)} ([\mathsf{C}_1]^{\mathsf{b}} b_0)(\mathtt{x}) &+ \\
\sum_{\mathtt{x} \in vars(\mathsf{C}_2) \smallsetminus vars(\mathsf{C}_1)} ([\mathsf{C}_2]^{\mathsf{b}} b_0)(\mathtt{x}) &= \\
[\mathsf{C}_1 + \mathsf{C}_2]^{\mathsf{b}} b_0
\end{aligned}$$

$(\mathsf{C}_1; \mathsf{C}_2)$ identical to $(\mathsf{C}_1 + \mathsf{C}_2)$;

(C^*)

$$\begin{aligned}
(\mathsf{C}^*)^{\mathsf{b}} &= \\
|vars(C) + 1|(\mathsf{C})^{\mathsf{b}} &= && \text{by inductive hypothesis} \\
|vars(C) + 1| \sum_{\mathtt{x} \in vars(\mathsf{C})} ([\mathsf{C}]^{\mathsf{b}} b_0)(\mathtt{x}) &= \\
\sum_{\mathtt{x} \in vars(\mathsf{C})} |vars(C) + 1|([\mathsf{C}]^{\mathsf{b}} b_0)(\mathtt{x}) &= \\
[\mathsf{fix}(\mathsf{C})]^{\mathsf{b}} b_0
\end{aligned}$$

□

We next prove an easy graph-theoretic property which will later be helpful. Consider a finite directed and edge-weighted graph $\langle X, \to \rangle$ where $\to \subseteq X \times \mathbb{Z} \times X$ and $x \to_h x'$ denotes that $(x, h, x') \in \to$. Consider a finite path in $\langle X, \to \rangle$

$$p = x_0 \to_{h_0} x_1 \to_{h_1} x_2 \to_{h_2} \dots \to_{h_{\ell-1}} x_\ell$$

where:

(i). $\ell \geq 1$

(ii). the carrier size of p is $s(p) \triangleq |\{x_0, ..., x_\ell\}|$

(iii). the weight of p is $w(p) \triangleq \Sigma_{k=0}^{\ell-1} h_k$

(iv). the length of p is $|p| \triangleq \ell$

(v). given indices $0 \leqslant i < j \leqslant \ell$, $p_{i,j}$ denotes the subpath of p given by $x_i \to_{h_i} x_{i+1} \to_{i+1} \cdots \to_{h_{j-1}} x_j$ whose length is $j - i$; $p_{i,j}$ is a cycle if $x_i = x_j$.

Lemma 2.4 (**Positive cycles in weighted directed graphs**). *Let p be a finite path*

$$p = x_0 \to_{h_0} x_1 \to_{h_1} x_2 \to_{h_2} \cdots \to_{h_{\ell-1}} x_\ell$$

with $m \triangleq \max\{|h_j| \mid j \in \{0, \dots, \ell - 1\}\} \in \mathbb{N}$ and $w(p) > (|X| - 1)m$. Then, p has a subpath which is a cycle having a strictly positive weight.

Proof. First note that $w(p) = \Sigma_{k=0}^{\ell-1} h_k > (|X|-1)m$ implies that $|p| = \ell \geq |X|$. Then, we show our claim by induction on $|p| = \ell \geq |X|$.
($|p| = |X|$): Since the path p includes exactly $|X|+1 = \ell+1$ nodes, there exist indices $0 \leqslant i < j \leqslant \ell$ such that $x_i = x_j$, i.e., $p_{i,j}$ is a subpath of p which is a cycle. Moreover, since this cycle $p_{i,j}$ includes at least one edge, we have that

$$\begin{array}{rl} w(p_{i,j}) = w(p) - (\Sigma_{k=0}^{i-1} h_k + \Sigma_{k=j}^{\ell-1} h_k) > & [\text{as } w(p) > (|X|-1)m] \\ (|X|-1)m - (\Sigma_{k=0}^{i-1} h_k + \Sigma_{k=j}^{\ell-1} h_k) \geq & [\text{as } \Sigma_{k=0}^{i-1} h_k + \Sigma_{k=j}^{\ell-1} h_k \leqslant (\ell-1)m] \\ (|X|-1)m - (\ell-1)m = & [\text{as } \ell = |X|] \\ (|X|-1)m - (|X|-1)m = 0 & \end{array}$$

so that $w(p_{i,j}) > 0$ holds.

($|p| > |X|$): Since the path p includes at least $|X|+2$ nodes, as in the base case, we have that p has a subpath which is a cycle. Then, we consider a cycle $p_{i,j}$ in p, for some indices $0 \leqslant i < j \leqslant \ell$, which is maximal, i.e., such that if $p_{i',j'}$ is a cycle in p, for some $0 \leqslant i' < j' \leqslant \ell$, then $p_{i,j}$ is not a proper subpath of $p_{i',j'}$.
If $w(p_{i,j}) > 0$ then we are done. Otherwise we have that $w(p_{i,j}) \leqslant 0$ and we consider the path p' obtained from p by stripping off the cycle $p_{i,j}$, i.e.,

$$p' \equiv \overbrace{x_0 \to_{h_0} x_1 \to_{h_1} \cdots \to_{h_{i-1}} x_i}^{p'_{0,i}} = \overbrace{x_j \to_{h_{j+1}} \cdots \to_{h_{\ell-1}} x_\ell}^{p'_{j+1,\ell}}$$

Since $|p'| < |p|$ and $w(p') = w(p) - w(p_{i,j}) \geq w(p) > (|X|-1)m$, we can apply the inductive hypothesis on p'. We therefore derive that p' has a subpath q which is a cycle having strictly positive weight. This cycle q is either entirely in $p'_{0,i}$ or in $p'_{j+1,\ell}$, otherwise q would include the cycle $p_{i,j}$ thus contradicting the maximality of $p_{i,j}$. Hence, q is a cycle in the original path p having a strictly positive weight. □

Lemma 2.5. *Let* $\mathsf{C} \in \mathit{Imp}$ *and* $\mathsf{y} \in \mathit{Var}$.
For all $\eta \in \mathbb{A}$ *and* $\mathsf{y} \in \mathit{Var}$, *if* $\max(\llbracket \mathsf{C} \rrbracket \eta \mathsf{y}) \neq \infty$ *and* $\max(\llbracket \mathsf{C} \rrbracket \eta \mathsf{y}) > (\mathsf{C})^{\mathsf{b}}$ *then there exist a variable* $\mathsf{z} \in \mathit{Var}$ *and an integer* $h \in \mathbb{Z}$ *such that* $|h| \leqslant (\mathsf{C})^{\mathsf{b}}$ *and the following two properties hold:*

i $\max(\llbracket \mathsf{C} \rrbracket \eta \mathsf{y}) = \max(\eta \mathsf{z}) + h$*;*

ii for all $\eta' \in \mathbb{A}$*, if* $\eta' \sqsupseteq \eta$ *then* $\max(\llbracket \mathsf{C} \rrbracket \eta' \mathsf{y}) \geq \max(\eta' \mathsf{z}) + h$*.*

Proof. The proof is by structural induction on the command $\mathsf{C} \in$ Imp. We preliminarily observe that we can safely assume $\eta \neq \bot$. In fact, if $\eta = \bot$ then $\llbracket \mathsf{C} \rrbracket \bot = \bot$ and thus $\max(\llbracket \mathsf{C} \rrbracket \eta \mathsf{y}) = 0 \leqslant (\mathsf{C})^{\mathsf{b}}$, against the hypothesis $\max(\llbracket \mathsf{C} \rrbracket \eta \mathsf{y}) > (\mathsf{C})^{\mathsf{b}}$. Moreover, when quantifying over η' such that $\eta' \sqsupseteq \eta$ in (i), if $\max(\llbracket \mathsf{C} \rrbracket \eta' \mathsf{y}) = \infty$ holds, then $\max(\llbracket \mathsf{C} \rrbracket \eta' \mathsf{y}) \geq \max(\eta' \mathsf{z}) + h$ trivially holds, hence we will sometimes silently omit to consider this case.

Case ($\mathsf{x} \in S$)
Take $\eta \in \mathbb{A}$ and assume $\infty \neq \max(\llbracket \mathsf{x} \in S \rrbracket \eta \mathsf{y}) > (\mathsf{x} \in S)^{\mathsf{b}}$. Clearly $\llbracket \mathsf{x} \in S \rrbracket \eta \neq \bot$, otherwise we would get the contradiction $\max(\llbracket \mathsf{x} \in S \rrbracket \eta \mathsf{y}) = 0 \leqslant (\mathsf{x} \in S)^{\mathsf{b}}$.

We distinguish two cases:

- If $\mathsf{y} \neq \mathsf{x}$, then for all $\eta' \in \mathbb{A}$ such that $\eta \sqsubseteq \eta'$ it holds $\bot \neq \llbracket \mathsf{x} \in S \rrbracket \eta' = \eta'[\mathsf{x} \mapsto \eta(\mathsf{x}) \sqcap \mathit{Int}(S)]$ and thus
$$\max(\llbracket \mathsf{x} \in S \rrbracket \eta' \mathsf{y}) = \max(\eta' \mathsf{y}) = \max(\eta' \mathsf{y}) + 0$$
hence the thesis follows with $\mathsf{z} = \mathsf{y}$ and $h = 0$.

- If $\mathsf{y} = \mathsf{x}$ then $\eta(\mathsf{x}) \in \mathit{Int}_*$ and
$$\max(\llbracket \mathsf{x} \in S \rrbracket \eta \mathsf{y}) = \max(\eta(\mathsf{x}) \sqcap \mathit{Int}(S))$$
Note that it cannot be $\max(S) \in \mathbb{N}$. Otherwise, by Definition 2.9, $\max(\eta(\mathsf{x}) \sqcap \mathit{Int}(S)) \leqslant \max(S) = (\mathsf{x} \in S)^{\mathsf{b}}$, violating the assumption $\max(\llbracket \mathsf{x} \in S \rrbracket \eta \mathsf{y}) > (\mathsf{x} \in S)^{\mathsf{b}}$. Hence, $\max(S) =$

∞ must hold and therefore $\max(\eta(\mathtt{x}) \sqcap \mathit{Int}(S)) = \max(\eta(\mathtt{x})) = \max(\eta(\mathtt{x}))+0$. It is immediate to check that the same holds for all $\eta' \sqsupseteq \eta$, i.e.,

$$\max(\eta'(\mathtt{x}) \sqcap \mathit{Int}(S)) = \max(\eta'(\mathtt{x})) = \max(\eta'(\mathtt{x})) + 0$$

and thus the thesis follows with $\mathtt{z} = \mathtt{y} = \mathtt{x}$ and $h = 0$.

Case (true) A consequence of the fact that $\mathsf{true} \equiv x \in \mathbb{N}$.

Case (false) A consequence of the fact that $\mathsf{false} \equiv x \in \varnothing$.

Case ($\mathtt{x} := k$) Take $\eta \in \mathbb{A}$ and assume $\max([\![\mathtt{x} := k]\!]\eta\mathtt{y}) > (\mathtt{x} := k)^{\mathsf{b}} = k$.

Observe that it cannot be $\mathtt{x} = \mathtt{y}$. In fact, since $[\![\mathtt{x} := k]\!]\eta = \eta[\mathtt{x} \mapsto [k,k]]$, we would have $[\![\mathtt{x} := k]\!]\eta\mathtt{y} = [k,k]$ and thus

$$\max([\![\mathtt{x} := k]\!]\eta\mathtt{y}) = k = (\mathtt{x} := k)^{\mathsf{b}}.$$

violating the assumption. Therefore, it must be $\mathtt{y} \neq \mathtt{x}$. Now, for all $\eta' \sqsupseteq \eta$, we have $[\![\mathtt{x} := k]\!]\eta'\mathtt{y} = \eta'\mathtt{y}$ and thus

$$\max([\![\mathtt{x} := k]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{y}) = \max(\eta'\mathtt{y}) + 0,$$

hence the thesis holds with $h = 0 \leqslant (\mathtt{x} := k)^{\mathsf{b}}$ and $\mathtt{z} = \mathtt{y}$.

Case ($\mathtt{x} := \mathtt{w} + k$) Take $\eta \in \mathbb{A}$ and assume $\max([\![\mathtt{x} := \mathtt{w} + k]\!]\eta\mathtt{y}) > (\mathtt{x} := \mathtt{w} + k)^{\mathsf{b}} = k$. Recall that $[\![\mathtt{x} := \mathtt{w} + k]\!]\eta = \eta[\mathtt{x} \mapsto \eta\mathtt{w} + k]$.

We distinguish two cases:

- If $\mathtt{y} \neq \mathtt{x}$, then for all $\eta' \sqsupseteq \eta$, we have $[\![\mathtt{x} := \mathtt{w} + k]\!]\eta'\mathtt{y} = \eta'\mathtt{y}$ and thus

  $$\max([\![\mathtt{x} := \mathtt{w} + k]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{y}).$$

  hence the thesis follows with $h = 0 \leqslant (\mathtt{x} := \mathtt{w} + k)^{\mathsf{b}}$ and $\mathtt{z} = \mathtt{y}$.

- If $\mathtt{x} = \mathtt{y}$ then for all $\eta' \sqsupseteq \eta$, we have $[\![\mathtt{x} := \mathtt{w} + k]\!]\eta'\mathtt{y} = \eta'\mathtt{w} + k$ and thus

  $$\max([\![\mathtt{x} := \mathtt{w} + k]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{w}) + k.$$

  Hence, the thesis follows with $h = k \leqslant (\mathtt{x} := \mathtt{w} + k)^{\mathsf{b}}$ and $\mathtt{z} = \mathtt{w}$.

Case ($\mathtt{x} := \mathtt{w} - k$) Take $\eta \in \mathbb{A}$ and assume $\max([\![\mathtt{x} := \mathtt{w} - k]\!]\eta\mathtt{y}) > (\mathtt{x} := \mathtt{w} - k)^{\mathsf{b}} = k$. Recall that $[\![\mathtt{x} := \mathtt{w} - k]\!]\eta = \eta[\mathtt{x} \mapsto \eta\mathtt{w} - k]$.

We distinguish two cases:

- If $\mathtt{y} \neq \mathtt{x}$, then for all $\eta' \in \mathbb{A}$ such that $\eta \sqsubseteq \eta'$, we have $[\![\mathtt{x} := \mathtt{w} - k]\!]\eta'\mathtt{y} = \eta'\mathtt{y}$ and thus

  $$\max([\![\mathtt{x} := \mathtt{w} - k]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{y}).$$

  hence the thesis holds, with $h = 0 \leqslant (\mathtt{x} := \mathtt{w} - k)^{\mathsf{b}}$ and $\mathtt{z} = \mathtt{y}$.

- If $\mathtt{x} = \mathtt{y}$ then for all $\eta' \in \mathbb{A}$ such that $\eta \sqsubseteq \eta'$, we have $[\![\mathtt{x} := \mathtt{w} - k]\!]\eta'\mathtt{y} = \eta'\mathtt{w} - k$ and thus

  $$\max([\![\mathtt{x} := \mathtt{w} - k]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{w}) - k.$$

  Note that the assumption $\max([\![\mathtt{x} := \mathtt{w} - k]\!]\eta\mathtt{y}) > k$ and thus $\max([\![\mathtt{x} := \mathtt{w} - k]\!]\eta'\mathtt{y}) > k$ ensures that subtraction is not truncated on the maximum.

  Hence the thesis holds, with $h = -k$, hence $|h| = (\mathtt{x} := \mathtt{w} - k)^{\mathsf{b}}$, and $\mathtt{z} = \mathtt{w}$.

Case $(\mathsf{C}_1 + \mathsf{C}_2)$ Take $\eta \in \mathbb{A}$ and assume $\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta) > (\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$.

Recall that $\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta = \llbracket \mathsf{C}_1 \rrbracket \eta \sqcup \llbracket \mathsf{C}_2 \rrbracket \eta$. Hence, since $\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \mathtt{y} \neq \infty$, we have that $\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y} \neq \infty \neq \llbracket \mathsf{C}_2 \rrbracket \eta \mathtt{y}$.

Moreover

$$\begin{aligned}\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \mathtt{y}) &= \max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y} \sqcup \llbracket \mathsf{C}_2 \rrbracket \eta \mathtt{y}) \\ &= \max\{\max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y}), \max(\llbracket \mathsf{C}_2 \rrbracket \eta \mathtt{y})\}\end{aligned}$$

Thus $\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \mathtt{y}) = \max(\llbracket \mathsf{C}_i \rrbracket \eta \mathtt{y})$ for some $i \in \{1, 2\}$. We can assume, without loss of generality, that the maximum is realized by the first component, i.e., $\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \mathtt{y}) = \max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y})$. Hence, by inductive hypothesis on C_1, we have that there exists $h \in \mathbb{Z}$ with $|h| \leqslant (\mathsf{C}_1)^{\mathsf{b}}$ and $\mathtt{z} \in \mathit{Var}$ such that $\max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y}) = \max(\eta \mathtt{z}) + h$ and for all $\eta' \in \mathbb{A}$, $\eta \sqsubseteq \eta'$,

$$\max(\llbracket \mathsf{C}_1 \rrbracket \eta' \mathtt{y}) \geq \max(\eta' \mathtt{z}) + h$$

Therefore

$$\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \mathtt{y}) = \max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{y}) = \max(\eta \mathtt{z}) + h$$

and and for all $\eta' \in \mathbb{A}$, $\eta \sqsubseteq \eta'$,

$$\begin{aligned}\max(\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta' \mathtt{y}) &= \max\{\max(\llbracket \mathsf{C}_1 \rrbracket \eta' \mathtt{y}), \max(\llbracket \mathsf{C}_2 \rrbracket \eta' \mathtt{y})\} \\ &\geq \max(\llbracket \mathsf{C}_1 \eta' \rrbracket \mathtt{y}) \\ &\geq \max(\eta' \mathtt{z}) + h\end{aligned}$$

with $|h| \leqslant (\mathsf{C}_1)^{\mathsf{b}} \leqslant (\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}}$, as desired.

Case $(\mathsf{C}_1; \mathsf{C}_2)$ Take $\eta \in \mathbb{A}$ and assume $\max(\llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta) > (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$.

Recall that $\llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta = \llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta)$. If we define

$$\llbracket \mathsf{C}_1 \rrbracket \eta = \eta_1$$

since $\max(\mathsf{C}_2 \eta_1 \mathtt{y}) \neq \infty$ and $\max(\mathsf{C}_2 \eta_1 \mathtt{y}) > (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} \geq (\mathsf{C}_2)^{\mathsf{b}}$, by inductive hypothesis on C_2, there are $|h_2| \leqslant (\mathsf{C}_2)^{\mathsf{b}}$ and $\mathtt{w} \in \mathit{Var}$ such that $\max(\llbracket \mathsf{C}_2 \rrbracket \eta_1 \mathtt{y}) = \max(\eta_1 \mathtt{w}) + h_2$ and for all $\eta'_1 \in \mathbb{A}$ with $\eta_1 \sqsubseteq \eta'_1$

$$\max(\llbracket \mathsf{C}_2 \rrbracket \eta'_1 \mathtt{y}) \geq \max(\eta'_1 \mathtt{w}) + h_2 \tag{†}$$

Now observe that $\max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{w}) = \max(\eta_1 \mathtt{w}) > (\mathsf{C}_1)^{\mathsf{b}}$. Otherwise, if it were $\max(\eta_1 \mathtt{w}) \leqslant (\mathsf{C}_1)^{\mathsf{b}}$ we would have

$$\max(\llbracket \mathsf{C}_2 \rrbracket \eta_1 \mathtt{y}) = \max(\eta_1 \mathtt{w}) + h_2 \leqslant (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}},$$

violating the hypotheses. Moreover, $\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{w} \neq \infty$, otherwise we would have $\max(\llbracket \mathsf{C}_2 \rrbracket \eta_1 \mathtt{y}) = \max(\eta_1 \mathtt{w}) + h_2 = \infty$, contradicting the hypotheses. Therefore we can apply the inductive hypothesis also to C_1 and deduce that there are $|h_1| \leqslant (\mathsf{C}_1)^{\mathsf{b}}$ and $\mathtt{w}' \in \mathit{Var}$ such that $\max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{w}) = \max(\eta \mathtt{w}') + h_1$ and for all $\eta' \in \mathbb{A}$ with $\eta \sqsubseteq \eta'$

$$\max(\llbracket \mathsf{C}_1 \rrbracket \eta' \mathtt{w}) \geq \max(\eta' \mathtt{w}') + h_1 \tag{‡}$$

Now, for all $\eta' \in \mathbb{A}$ with $\eta \sqsubseteq \eta'$ we have that:

$$\begin{aligned}\max(\llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta \mathtt{y}) &= \max(\llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta) \mathtt{y}) \\ &= \max(\llbracket \mathsf{C}_2 \rrbracket \eta_1 \mathtt{y}) \\ &= \max(\eta_1 \mathtt{w}) + h_2 \\ &= \max(\llbracket \mathsf{C}_1 \rrbracket \eta \mathtt{w}) + h_2 \\ &= \max(\eta \mathtt{w}') + h_1 + h_2\end{aligned}$$

and

$$
\begin{aligned}
\max(\llbracket \mathsf{C}_1; \mathsf{C}_2 \rrbracket \eta' \mathtt{y}) &= \\
\max(\llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta')\mathtt{w}) &\geq \\
\max(\llbracket \mathsf{C}_1 \rrbracket \eta' \mathtt{w}') + h_2 &\geq && \text{by } (\dagger)\text{, since } \eta_1 = \llbracket \mathsf{C}_1 \rrbracket \eta \sqsubseteq \llbracket \mathsf{C}_1 \rrbracket \eta' \text{ , by monotonicity} \\
(\max(\eta' \mathtt{y}) + h_1) + h_2 & && \text{by } (\ddagger)
\end{aligned}
$$

Thus, the thesis holds with $h = h_1 + h_2$, as $|h| = |h_1 + h_2| \leqslant |h_1| + |h_2| \leqslant (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}}$, as needed.

Case ($\mathsf{fix}(\mathsf{C})$) Let $\eta \in \mathbb{A}$ such that $\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y} \neq \infty$. Recall that $\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta = \mathrm{lfp} \lambda\mu.(\llbracket \mathsf{C} \rrbracket \mu \sqcup \eta)$. Observe that the least fixpoint of $\lambda\mu.(\llbracket \mathsf{C} \rrbracket \mu \sqcup \eta)$ coincides with the least fixpoint of $\lambda\mu.(\llbracket \mathsf{C} \rrbracket \mu \sqcup \mu) = \lambda\mu.\llbracket \mathsf{C} + \mathsf{true} \rrbracket \mu$ above η. Hence, if

- $\eta_0 \triangleq \eta$,
- for all $i \in \mathbb{N}$, $\eta_{i+1} \triangleq \llbracket \mathsf{C} \rrbracket \eta_i \sqcup \eta_i = \llbracket \mathsf{C} + \mathsf{true} \rrbracket \eta_i \sqsupseteq \eta_i$,

then we define an increasing chain $\{\eta_i\}_{i \in \mathbb{N}} \subseteq \mathbb{A}$ such that

$$\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta = \textstyle\bigsqcup_{i \in \mathbb{N}} \eta_i.$$

Since $\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y} \neq \infty$, we have that for all $i \in \mathbb{N}$, $\eta_i \mathtt{y} \neq \infty$. Moreover, the lub $\bigsqcup_{i \in \mathbb{N}} \eta_i$ on $\mathtt{y}$ is finitely reached in the chain $\{\eta_i\}_{i \in \mathbb{N}}$, i.e., there exists $m \in \mathbb{N}$ such that for all $i \geq m + 1$

$$\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y} = \eta_i \mathtt{y}.$$

The inductive hypothesis holds for C and true, hence for $\mathsf{C} + \mathsf{true}$, therefore for all $\mathtt{x} \in \mathit{Var}$ and $j \in \{0, 1, \ldots, m\}$, if $\max(\eta_{j+1}\mathtt{x}) > (\mathsf{C} + \mathsf{true})^{\mathsf{b}} = (\mathsf{C})^{\mathsf{b}}$ then there exist $\mathtt{z} \in \mathit{Var}$ and $h \in \mathbb{Z}$ such that $|h| \leqslant (\mathsf{C})^{\mathsf{b}}$ and

(a) $\infty \neq \max(\eta_{j+1}\mathtt{x}) = \max(\eta_j \mathtt{z}) + h$,

(b) $\forall \eta' \sqsupseteq \eta_j.\, \max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket \eta' \mathtt{x}) \geq \max(\eta' \mathtt{z}) + h$.

To shortly denote that the two conditions (a) and (b) hold, we write

$$(\mathtt{z}, j) \to_h (\mathtt{x}, j + 1)$$

Now, assume that for some variable $\mathtt{y} \in \mathit{Var}$

$$\max(\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y}) = \max(\eta_{m+1}\mathtt{y}) > (\mathsf{fix}(\mathsf{C}))^{\mathsf{b}} = (n + 1)(\mathsf{C})^{\mathsf{b}}$$

where $n = |\mathit{vars}(\mathsf{C})|$. We want to show that the thesis holds, i.e., that there exist $\mathtt{z} \in \mathit{Var}$ and $h \in \mathbb{Z}$ with $|h| \leqslant (\mathsf{fix}(\mathsf{C}))^{\mathsf{b}}$ such that:

$$\max(\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y}) = \max(\eta \mathtt{z}) + h \tag{i}$$

and for all $\eta' \sqsupseteq \eta$,

$$\max(\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta' \mathtt{y}) \geq \max(\eta' \mathtt{z}) + h \tag{ii}$$

Let us consider (i). We first observe that we can define a path

$$\sigma \triangleq (\mathtt{y}_0, 0) \to_{h_0} (\mathtt{y}_1, 1) \to_{h_1} \ldots \to_{h_m} (\mathtt{y}_{m+1}, m + 1) \tag{2.2}$$

such that $\mathtt{y}_{m+1} = \mathtt{y}$ and for all $j \in \{0, \ldots, m + 1\}$, $\mathtt{y}_j \in \mathit{Var}$ and $\max(\eta_j \mathtt{y}_j) > (C)^{\mathsf{b}}$. In fact, if, by contradiction, this is not the case, there would exist an index $i \in \{0, \ldots, m\}$ (as $\max(\eta_{m+1}\mathtt{y}_{m+1}) > (C)^{\mathsf{b}}$ already holds) such that $\max(\eta_i \mathtt{y}_i) \leqslant (C)^{\mathsf{b}}$, while for all $j \in \{i + 1, \ldots, m + 1\}$, $\max(\eta_j \mathtt{y}_j) > (C)^{\mathsf{b}}$. Thus, in such a case, we consider the nonempty path:

$$\pi \triangleq (\mathtt{y}_i, i) \to_{h_i} (\mathtt{y}_{i+1}, i + 1) \to_{h_{i+1}} \cdots \to_{h_m} (\mathtt{y}_{m+1}, m + 1)$$

and we have that:

$$\begin{aligned}
&\Sigma_{j=i}^{m} h_j = \\
&\Sigma_{j=i}^{m} \max(\eta_{j+1}\mathtt{y}_{j+1}) - \max(\eta_j\mathtt{y}_j) = \\
&\max(\eta_{m+1}\mathtt{y}_{m+1}) - \max(\eta_i\mathtt{y}_i) = \\
&\max(\eta_{m+1}\mathtt{y}) - \max(\eta_i\mathtt{y}_i) > \\
&(n+1)(\mathsf{C})^{\mathsf{b}} - (\mathsf{C})^{\mathsf{b}} = n(\mathsf{C})^{\mathsf{b}}
\end{aligned}$$

with $|h_j| \leqslant (\mathsf{C})^{\mathsf{b}}$ for $j \in \{i, \ldots, m\}$. Hence we can apply Lemma 2.4 to the projection π_p of the nodes of this path π to the variable component to deduce that π_p has a subpath which is a cycle with a strictly positive weight. More precisely, there exist $i \leqslant k_1 < k_2 \leqslant m+1$ such that $\mathtt{y}_{k_1} = \mathtt{y}_{k_2}$ and $h = \Sigma_{j=k_1}^{k_2-1} h_j > 0$. If we denote $\mathtt{w} = \mathtt{y}_{k_1} = \mathtt{y}_{k_2}$, then we have that

$$\begin{aligned}
\max(\eta_{k_2}\mathtt{w}) &= h_{k_2-1} + \max(\eta_{k_2-1}\mathtt{w}) \\
&= h_{k_2-1} + h_{k_2-2} + \max(\eta_{k_2-2}\mathtt{w}) \\
&= \Sigma_{j=k_1}^{k_2-1} h_j + \max(\eta_{k_1}\mathtt{w}) \\
&= h + \max(\eta_{k_1}\mathtt{w})
\end{aligned}$$

Thus,

$$\max([\![\mathsf{C} + \mathsf{true}]\!]^{k_2-k_1}\eta_{k_1}\mathtt{w}) = \max(\eta_{k_1}\mathtt{w}) + h$$

Observe that for all $\eta' \sqsupseteq \eta_{k_1}$

$$\max([\![\mathsf{C} + \mathsf{true}]\!]^{k_2-k_1}\eta'\mathtt{w}) \geq \max(\eta'\mathtt{w}) + h \tag{2.3}$$

This property (2.3) can be shown by induction on $k_2 - k_1 \geq 1$.
Then, an inductive argument allows us to show that for all $r \in \mathbb{N}$:

$$\max([\![\mathsf{C} + \mathsf{true}]\!]^{r(k_2-k_1)}\eta_{k_1}\mathtt{w}) \geq \max(\eta_{k_1}\mathtt{w}) + rh \tag{2.4}$$

In fact, for $r = 0$ the claim trivially holds. Assuming the validity for $r \geq 0$ then we have that

$$\begin{aligned}
&\max([\![\mathsf{C} + \mathsf{true}]\!]^{(r+1)(k_2-k_1)}\eta_{k_1}\mathtt{w}) = \\
&\max([\![\mathsf{C} + \mathsf{true}]\!]^{k_2-k_1}([\![\mathsf{C} + \mathsf{true}]\!]^{r(k_2-k_1)}\eta_{k_1})\mathtt{w}) \geq \quad [\text{by (2.3) as } \eta_{k_1} \sqsubseteq [\![\mathsf{C} + \mathsf{true}]\!]^{r(k_2-k_1)}\eta_{k_1}\ ] \\
&\max([\![\mathsf{C} + \mathsf{true}]\!]^{r(k_2-k_1)}\eta_{k_1}\mathtt{w}) + h \geq \quad [\text{by inductive hypothesis}] \\
&\max(\eta_{k_1}\mathtt{w}) + rh + h \geq \max(\eta_{k_1}\mathtt{w}) + (r+1)h
\end{aligned}$$

However, This would contradict the hypothesis $[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} \neq \infty$. In fact the inequality (2.4) would imply

$$\begin{aligned}
[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{w} &= \bigsqcup_{i\in\mathbb{N}} [\![\mathsf{C} + \mathsf{true}]\!]^{i}\eta\mathtt{w} = \\
&= \bigsqcup_{i\in\mathbb{N}} [\![\mathsf{C} + \mathsf{true}]\!]^{i}\eta_{k_1}\mathtt{w} \\
&= \bigsqcup_{r\in\mathbb{N}} [\![\mathsf{C} + \mathsf{true}]\!]^{r(k_2-k_1)}\eta_{k_1}\mathtt{w} \\
&= \infty
\end{aligned}$$

Now, from (2.2) we deduce that for all $\eta' \sqsupseteq \eta_{k_1}$, for $j \in \{k_1, \ldots, m\}$, if we let $\mu_{k_1} = \eta'$ and $\mu_{j+1} = [\![\mathsf{C} + \mathsf{true}]\!]\mu_j$, we have that $\max(\mu_{j+1}\mathtt{y}_{j+1}) \geq \max(\mu_{j+1}\mathtt{y}_j) + h_j$ and thus

$$[\![\mathsf{C} + \mathsf{true}]\!]^{m-k_1+1}\eta'\mathtt{y} = \mu_{m+1}\mathtt{y}_{m+1} \geq \max(\mathtt{y}_{k_1}) + \Sigma_{i=k_1}^{m} h_i = \max(\eta'\mathtt{w}) + \Sigma_{i=k_1}^{m} h_i$$

Since $\eta' = [\![\mathsf{fix}(\mathsf{C})]\!]\eta \sqsupseteq \eta_{k_1}$ we conclude

$$\begin{aligned}[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} &= [\![\mathsf{C} + \mathsf{true}]\!]^{m-k_1+1}[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} \\ &\geq \infty + \Sigma_{i=k_1}^{m} h_i = \infty\end{aligned}$$

contradicting the assumption.
Therefore, the path σ of (2.2) must exist, and consequently

$$\max([\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y}) = \max(\eta_{m+1}\mathtt{y}) = \max(\eta\mathtt{y}_0) + \Sigma_{i=0}^{m} h_i$$

and $\Sigma_{i=0}^{m} h_i \leqslant (\mathsf{fix}(\mathsf{C}))^{\mathsf{b}} = (n+1)(\mathsf{C})^{\mathsf{b}}$, otherwise we could use the same argument above for inferring the contradiction $p[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} = \infty$.

Let us now show (ii). Given $\eta' \sqsupseteq \eta$ from (2.2) we deduce that for all $j \in \{0, \ldots, m\}$, if we let $\mu_0 = \eta'$ and $\mu_{j+1} = [\![\mathsf{C} + \mathsf{true}]\!]\mu_j$, we have that

$$\max(\mu_{j+1}\mathtt{y}_{j+1}) \geq \max(\mu_{j+1}\mathtt{y}_j) + h_j.$$

Therefore, since $[\![\mathsf{fix}(\mathsf{C})]\!]\eta' \sqsupseteq \mu_{m+1}$ (observe that the convergence of $[\![\mathsf{fix}(\mathsf{C})]\!]\eta'$ could be at an index greater than $m+1$), we conclude that:

$$\max([\![\mathsf{fix}(\mathsf{C})]\!]\eta'\mathtt{y}) \geq \max(\mu_{m+1}\mathtt{y}) = \max(\mu_{m+1}\mathtt{y}_{m+1}) \geq \max(\eta'\mathtt{y}_0) + \Sigma_{i=0}^{m} h_i$$

as desired. □

Lemma 2.5 provides an effective algorithm for computing the interval semantics of commands. More precisely, given a command C, the corresponding finite set of variables $\mathit{Var}_{\mathsf{C}} \triangleq \mathit{vars}(\mathsf{C})$, and an interval environment $\rho : \mathit{Var}_{\mathsf{C}} \to \mathit{Int}$, we define

$$\max(\rho) \triangleq \max\{\max(\rho(\mathtt{x})) \mid \mathtt{x} \in \mathit{Var}_{\mathsf{C}}\}.$$

Then, when computing $\langle \mathsf{C}^* \rangle \rho$ on such ρ having a finite domain, we can restrict to a bounded interval domain $\mathbb{A}_{\mathsf{C},\rho} \triangleq (\mathit{Var}_{\mathsf{C}} \to \mathit{Int}_{\mathsf{C},\rho}) \cup \{\top, \bot\}$ where

$$\mathit{Int}_{\mathsf{C},\rho} \triangleq \{[a,b] \mid a, b \in \mathbb{N} \ \wedge \ a \leqslant b \leqslant \max\{\max(\rho), 2(\mathsf{C})^{\mathsf{b}}\}\}.$$

Lemma 2.6. *Let $\mathsf{C} \in \mathit{Imp}$ be a command. Then, for all finitely supported $\rho : \mathit{Var} \to \mathit{Int}$, the abstract semantics $\langle \mathsf{C}^* \rangle \rho = \bigsqcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i(\rho)$ computed in $\mathbb{A}$ and in $\mathbb{A}_{\mathsf{C},\rho}$ coincide.*

Proof. Todo: consequence of Lemma 2.5. □

Chapter 3

Non relational collecting

Let

$$\mathsf{Env}^{\mathsf{c}} \triangleq \{\eta \mid \eta : \mathit{Var} \to 2^{\mathbb{Z}} \setminus \{\varnothing\}\} \cup \{\bot\}.$$

The nonrelational collecting domain is the complete lattice $\mathbb{C}^{\mathsf{c}} \triangleq \langle \mathsf{Env}^{\mathsf{c}}, \dot{\subseteq} \rangle$ where for all $\eta, \eta' : \mathit{Var} \to \wp^{\neq\varnothing}(\mathbb{Z})$

$$\begin{aligned} &\bot \mathrel{\dot{\subseteq}} \eta \\ &\eta \mathrel{\dot{\subseteq}} \eta' \quad \text{if} \quad \forall \mathtt{x} \in \mathit{Var}.\eta(\mathtt{x}) \subseteq \eta'(\mathtt{x}) \end{aligned}$$

The nonrelation abstraction $\alpha : \langle 2^{\mathsf{Env}}, \subseteq \rangle \to \langle \mathsf{Env}^{\mathsf{c}}, \dot{\subseteq} \rangle$ is defined as follows:

$$\alpha(X) \triangleq \begin{cases} \bot & \text{if } X = \varnothing \\ \lambda\mathtt{x}.\{\rho(\mathtt{x}) \in \mathbb{Z} \mid \rho \in X\} & \text{if } X \neq \varnothing \end{cases}$$

while the concretization $\gamma : \langle \mathsf{Env}^{\mathsf{c}}, \dot{\subseteq} \rangle \to \langle 2^{\mathsf{Env}}, \subseteq \rangle$ is defined as follows:

$$\begin{aligned} &\gamma(\bot) \triangleq \varnothing \\ &\gamma(\eta) \triangleq \{\rho : \mathit{Var} \to \mathbb{Z} \mid \forall \mathtt{x} \in \mathit{Var}.\rho(\mathtt{x}) \in \eta(\mathtt{x})\} \end{aligned}$$

Bibliography

[Cut80] Nigel Cutland. *Computability: An introduction to recursive function theory.* Cambridge university press, 1980.

[Koz97] Dexter Kozen. "Kleene Algebra with Tests". In: *ACM Trans. Program. Lang. Syst.* 19.3 (May 1997), pp. 427–443. ISSN: 0164-0925. DOI: `10.1145/256167.256195`. URL: `https://doi.org/10.1145/256167.256195`.

[Ric53] Henry Gordon Rice. "Classes of recursively enumerable sets and their decision problems". In: *Transactions of the American Mathematical society* 74.2 (1953), pp. 358–366.