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

This thesis explores program verification trough abstract interpretation in the context of computability theory. Abstract Interpretation is a program analysis technique, based on approximating the semantics of programs over so-called *abstract domains*, usually represented as complete lattices, whose elements represent program properties. These approximations rely on some abstract operators, which usually include fixpoint iterations. Traditionally, to ensure convergence of such iterations, and therefore ensuring the termination of the analyzer, the literature relied on two important operators: the *widening* and the *narrowing* operators, first defined in [CC77]: the first one to compute an upper bound on some chain in the complete lattice, and the second one to recover some additional information from the program and refine the upper bound provided by the widening. This thesis focuses on a special abstract domain, called the *intervals* domain, where each variable of program is assigned to an interval over the integer numbers. The thesis argues that in such a context widening and narrowing operators can be replaced by another method, that relies on deciding program divergence by looking at the behavior of variables in the context of the program.

Acknowledgments

?

Contents

Chapter 1

Background

1.1 Previous work

[Gaw+09] was the first work to introduce a method to compute least solutions in polynomial time. It relies on a generalization of the Bellman-Ford algorithm from [Bel58] to find a lest solution for system of equations with addition and least upper bound. The method is then extended until the authors build a cubic time algorithm for the class of interval equations (equations with variables in the interval domain).

1.2 Recursion theory

Following [CGR18], we'll provide the general terminology and notation fr computable functions in recursion theory, as in [Cut80; Odi92; Rog87].

Definition 1.1 (Total functions). Let X, Y be two sets. Then

$$X \to Y$$

is the set of all total functions from X to Y.

Definition 1.2 (Partial functions). Let X, Y be two sets. Then

$$X \hookrightarrow Y$$

is the set of all partial functions from X to Y.

Partial functions are actually functions from a subset $S \subseteq X$ which is called the *natural domain* of f

Definition 1.3 (Domain of partial functions). Let $f : X \hookrightarrow Y$. $f(x) \downarrow$ means that f is defined on x, $f(x) \uparrow$ means that f is undefined on x. Hence

$$dom(f) = \{x \in X \mid f(x) \downarrow\}$$

We then need, mostly in section 2.4 to talk about partial recursive functions and their properties, namely the properties and conclusion that we derive from well known computability results. We therefore define partial recursive functions as follows:

Notation 1.1 (Set of partial recursive functions). By $\mathbb{N}^k \xrightarrow{r} \mathbb{N}$ we denote the set of partial recursive functions on natural numbers

Notation 1.2 (Set of total recursive functions). By $\mathbb{N} \xrightarrow{r} \mathbb{N}$ we denote the set of partial recursive functions on natural numbers

We also need to talk about decidable properties and decidable sets. We therefore introduce the notion of recursive and recursively enumerable sets.

Definition 1.4 (Recursively enumerable set). $A \subseteq \mathbb{N}^k$ is *recursively enumerable* (r.e. or semi-decidable) if $A = dom(f)$ for some $f \in \mathbb{N}^k \xrightarrow{r} \mathbb{N}$

Definition 1.5 (Recursive set). $A \subseteq \mathbb{N}$ is a recursive set if both A and its complement $\overline{A} = \mathbb{N} \setminus A$ are semi-decidable, i.e., there exists some $f \in \mathbb{N} \xrightarrow{r} \mathbb{N}$ s.t.

$$f = \lambda n.(n \in A)?1 : 0$$

Lemma 1.1 (Computable function over a recursive set). *Given $f : A \xrightarrow{r} B$, let the domain A to be recursive. B is at least r. e.*

Proof. $f : A \xrightarrow{r} B$ total recursive function over a recursive set A. We can write the function

$$\mathcal{X}_B = \lambda x.sg(\mu z.|f(z) - x|)$$

which is computable as it is composition of computable functions. In other terms, this function

$$\mathcal{X}_B(x) = \begin{cases} 1 & x \in B \\ \uparrow & \text{otherwise} \end{cases}$$

is the semi-decision function for B □

Observation 1.1. In general, B is not recursive, as it would means that both $A \leqslant_m B$ and $B \leqslant_m A$, which is not always the case, but it is r.e., as we could always write the inverse function as in lemma 1.1 and derive a semi-decision function for the image of the function.

1.3   Order theory

Within Theoretical Computer Science, especially in the field of semantics, partial orders hold significant importance. They are extensively employed in Abstract Interpretation, as highlighted in [Min18], serving different levels of the theory to model core notions. These notions include the idea of approximation, where certain analysis results may be less precise than others, creating a partial order where some results are incomparable. Moreover, partial orders are fundamental in conveying the concept of soundness: an analysis is deemed sound if its result is more general than the actual behavior. These mathematical notions, essential for discussions surrounding the Abstract Interpretation formalism, primarily involve order and lattice theory.

Definition 1.6 (Partiall ordered set). Let X be a non-empty set, $\sqsubseteq \subseteq X \times X$ be a reflexive, antisymmetric and transitive relation on that set, i.e., $\forall x, y, z \in X$:

1. $x \sqsubseteq x$ (reflexivity)

2. $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$ (antisymmetry)

3. $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$ (transitivity)

Then the tuple $\langle X, \sqsubseteq \rangle$ is a *partially ordered set* (POSet).

Definition 1.7 (Least upper bound). Let $\langle X, \sqsubseteq \rangle$ be a POSet and let $Z \subseteq X$. We say that $\overline{z}$ is an *upper bound* on Z if $\forall z \in Z \ z \sqsubseteq \overline{z}$. It is the *least upper bound* of Z (denoted as $\cup_X Z$) if $\forall z' \in Z$ upper bounds on Z, $\overline{z} \sqsubseteq z'$.

Definition 1.8 (Greatest lower bound). Let $\langle X, \sqsubseteq \rangle$ be a POSet and let $Z \subseteq X$. We say that $\overline{z}$ is a *lower bound* on Z if $\forall z \in Z \ \overline{z} \sqsubseteq z$. It is the *greatest lower bound* of Z (denoted as $\cap_X Z$) if $\forall z' \in Z$ upper bounds on Z, $z' \sqsubseteq \overline{z}$.

Usually then we're talking about least and greatest lower bound the bigger set is often implicit, and we therefore simply write $\cup Z$ and $\cap Z$. In abstract interpretation we often rely on special kinds of POSets, where the existance of the greatest lower bound and the least upper bound is ensured for each subset of the original POSet. These sets are called complete lattices

Definition 1.9 (Complete lattice). A POSet $\langle X, \sqsubseteq \rangle$ is called a *complete lattice* if

$$\forall Y \subseteq X \quad \exists \cup Y \wedge \exists \cap Y$$

Complete lattices are a subset of the class of chain complete partial ordered sets. These kinds of partial orders are defined using the concept of chains:

Definition 1.10 (Chain). Let $\langle D, \sqsubseteq \rangle$ be a partially ordered set. Then $Y \subseteq D$ is a chain if for any $y_1, y_2 \in Y$ it holds that

$$y_1 \sqsubseteq y_2 \vee y_2 \sqsubseteq y_1$$

Definition 1.11. $\langle D, \sqsubseteq \rangle$ is a chain complete partially ordered set (ccpo) if every chain of D has a least upper bound.

The last building block we will use in the following chapters is the Kleene-Knaster-Tarski theorem. This theorem is a fundamental result in order theory and provides a powerful tool for analyzing and establishing the existence of fixed points in complete lattices. To state it we need to first link functions and order theory with some definitions

Definition 1.12 (Monotone functions). Let $\langle D, \sqsubseteq \rangle$ and $\langle D', \sqsubseteq' \rangle$ be complete lattices. The total function $f : D \to D'$ is *monotone* if

$$d_1 \sqsubseteq d_2 \Rightarrow f(d_1) \sqsubseteq' f(d_2)$$

Monotonicity however does not preserve upper bounds, just orders. In particular if we take a chain $Y \subseteq D$ of some ccpo $\langle D, \subseteq \rangle$ and some monotone function $f : D \to D$, in general $\sqcup \{f(d) \mid d \in Y\} \sqsubseteq f(\sqcup Y)$, but not $\sqcup \{f(d) \mid d \in Y\} = f(\sqcup Y)$. Therefore we introduce the concept of continuity, functions that preserve both order and upper bounds

Definition 1.13 (Continuous functions). Let $\langle X, \sqsubseteq \rangle$ and $\langle X', \sqsubseteq' \rangle$ be ccpos. The total function $f : D \to D'$ is *continuous* if

- f is monotone;

- $\sqcup' \{f(d) \mid d \in D\} = f(\sqcup X)$

Continuous functions over ccpos are important for the Kleene fixed-point theorem, usually attributed to Tarski from [Tar55], which is also called kleene iteration. It gives us an iteration strategy to find the least fixpoint of a function over a ccpo, provided that the function is continuous.

Theorem 1.1 (Kleene fixed-point). *Let $f : D \to D$ be a continuous function over a chain complete partial order $\langle D, \sqsubseteq \rangle$ with the lest element $\bot$. Then*

$$\mathsf{fix}(f) = \sqcup \{f^n(\bot) \mid n \in \mathbb{N}\}$$

where

- $f^0 = id$

- $f^{n+1} = f \circ f^n \quad \forall n \in \mathbb{N}$

is the least fix point of f.

Chapter 2

Framework

2.1 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.

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

where $\mathrm{x}, \mathrm{y} \in \mathit{Var}$ 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.

2.2 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 2.1 (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:

$$(\!|\mathrm{x} \in S|\!)\rho \triangleq \begin{cases} \rho & \rho(\mathrm{x}) \in S \\ \bot & \text{otherwise} \end{cases}$$

$$(\!|\mathsf{true}|\!)\rho \triangleq \rho$$

$$(\!|\mathsf{false}|\!)\rho \triangleq \bot$$

$$(\!|\mathrm{x} := k|\!)\rho \triangleq \rho[\mathrm{x} \mapsto k]$$

$$(\!|\mathrm{x} := \mathrm{y} + k|\!)\rho \triangleq \begin{cases} \rho[\mathrm{x} \mapsto \rho(\mathrm{y}) + k] & \rho \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

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 2.2 (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 $\langle \cdot \rangle : \mathsf{Imp} \to \mathbb{C} \to \mathbb{C}$ which maps each program $\mathsf{C} \in \mathsf{Imp}$ to a total function over the complete lattice $\mathbb{C}$, inductively defined as follows: given $X \in \mathbb{C}$

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

We observe that the semantics we described is additive:

Observation 2.1 (Semantics Additivity)**.** Given $\mathsf{C} \in \mathsf{Imp}$, $X, Y \in \mathbb{C}$,

$$\langle \mathsf{C} \rangle (X \cup Y) = \langle \mathsf{C} \rangle X \cup \langle \mathsf{C} \rangle Y$$

Proof. We will prove it by induction on the program C:

Base case:

- $\mathsf{C} \equiv \mathsf{e}$ therefore

$$\begin{aligned} \langle \mathsf{e} \rangle (X \cup Y) &= \{ (\!| \mathsf{e} |\!) \rho \mid \rho \in X \cup Y, (\!| \mathsf{e} |\!) \rho \neq \bot \} \\ &= \{ (\!| \mathsf{e} |\!) \rho \mid \rho \in X \vee \rho \in Y, (\!| \mathsf{e} |\!) \rho \neq \bot \} \\ &= \{ (\!| \mathsf{e} |\!) \rho \mid \rho \in X, (\!| \mathsf{e} |\!) \rho \neq \bot \} \cup \{ (\!| \mathsf{e} |\!) \rho \mid \rho \in Y, (\!| \mathsf{e} |\!) \rho \neq \bot \} \\ &= \langle \mathsf{e} \rangle X \cup \langle \mathsf{e} \rangle Y \end{aligned}$$

Inductive cases:

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

$$\begin{aligned} \langle \mathsf{C}_1 + \mathsf{C}_2 \rangle (X \cup Y) &= \langle \mathsf{C}_1 \rangle (X \cup Y) \cup \langle \mathsf{C}_2 \rangle (X \cup Y) && \text{by definition} \\ &= \langle \mathsf{C}_1 \rangle X \cup \langle \mathsf{C}_1 \rangle Y \cup \langle \mathsf{C}_2 \rangle X \cup \langle \mathsf{C}_2 \rangle Y && \text{by inductive hypothesis} \\ &= \langle \mathsf{C}_1 + \mathsf{C}_2 \rangle X \cup \langle \mathsf{C}_1 + \mathsf{C}_2 \rangle Y \end{aligned}$$

- $\mathsf{C} \equiv \mathsf{C}_1 ; \mathsf{C}_2$ therefore

$$\begin{aligned} \langle \mathsf{C}_1 ; \mathsf{C}_2 \rangle (X \cup Y) &= \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle (X \cup Y)) && \text{by definition} \\ &= \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X \cup \langle \mathsf{C}_1 \rangle Y) && \text{by inductive hypothesis} \\ &= \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X) \cup \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle Y) && \text{by inductive hypothesis} \end{aligned}$$

- $\mathsf{C} \equiv \mathsf{C}^*$ therefore

$$\langle \mathsf{C}^* \rangle (X \cup Y) = \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i (X \cup Y)$$

in order to use the inductive hypothesis we have to show that

$$\forall i \in \mathbb{N} \quad \langle \mathsf{C} \rangle^i (X \cup Y) = \langle \mathsf{C} \rangle^i X \cup \langle \mathsf{C} \rangle^i Y$$

to do that, we work again by induction on i:

– the base case is $i = 0$ then $X \cup Y = X \cup Y$.

– For the inductive case we need to show that $i \Rightarrow i + 1$:

$$\begin{aligned}
\langle C \rangle^{i+1} (X \cup Y) &= \langle C \rangle \left(\langle C \rangle^i (X \cup Y) \right) \\
&= \langle C \rangle \left(\langle C \rangle^i X \cup \langle C \rangle^i Y \right) & \text{by induction hypothesis on } i \\
&= \langle C \rangle \left(\langle C \rangle^i X \right) \cup \langle C \rangle \left(\langle C \rangle^i Y \right) & \text{by induction hypothesis on } C \\
&= \langle C \rangle^{i+1} X \cup \langle C \rangle^{i+1} Y
\end{aligned}$$

Therefore we can use the inductive hypothesis internally and say

$$\begin{aligned}
\langle C^* \rangle (X \cup Y) &= \bigcup_{i \in \mathbb{N}} \langle C \rangle^i (X \cup Y) \\
&= \bigcup_{i \in \mathbb{N}} \left(\langle C \rangle^i X \cup \langle C \rangle^i Y \right) & \text{for the later statement} \\
&= \left(\bigcup_{i \in \mathbb{N}} \langle C \rangle^i X \right) \cup \left(\bigcup_{i \in \mathbb{N}} \langle C \rangle^i Y \right) \\
&= \langle C^* \rangle X \cup \langle C^* \rangle Y & \square
\end{aligned}$$

We can also observe that a program induces a monotone function in the concrete domain $\mathbb{C}$:

Lemma 2.1. *Given a program $C \in Imp$, the semantics function $\langle C \rangle : \mathbb{C} \to \mathbb{C}$ is monotone.*

Proof. We can prove this by induction on the program $C \in Imp$. Let $X, Y \in \mathbb{C}, X \subseteq Y$. We want to prove that $\langle C \rangle X \subseteq \langle C \rangle Y$.

Base case:

- $C \equiv e$ therefore

$$\begin{aligned}
\langle e \rangle X &= \{ (\!|e|\!) \rho \mid \rho \in X, (\!|e|\!)\rho \neq \bot \} \\
\langle e \rangle Y &= \{ (\!|e|\!) \rho \mid \rho \in Y, (\!|e|\!)\rho \neq \bot \}
\end{aligned}$$

$X \subseteq Y$ therefore $\rho \in X \Rightarrow \rho \in Y$ which also means that $\rho' \in \langle e \rangle X \Rightarrow \rho' \in \langle e \rangle Y$, therefore $\langle e \rangle X \subseteq \langle e \rangle Y$

Inductive cases:

- $C \equiv C_1 + C_2$ therefore we need to show that $\langle com_1 + C_2 \rangle X \subseteq \langle com_1 + C_2 \rangle Y$

$$\begin{aligned}
\langle C_1 + C_2 \rangle X &= \langle C_1 \rangle X \cup \langle C_2 \rangle X \\
\langle C_1 + C_2 \rangle Y &= \langle C_1 \rangle Y \cup \langle C_2 \rangle Y
\end{aligned}$$

by inductive hypothesis both $\langle C_1 \rangle X \subseteq \langle C_1 \rangle Y$ and $\langle C_2 \rangle X \subseteq \langle C_2 \rangle Y$ and therefore $\langle C_1 + C_2 \rangle X \subseteq \langle C_1 + C_2 \rangle Y$.

- $C \equiv C_1 ; C_2$ therefore we need to show that $\langle com_1 ; C_2 \rangle X \subseteq \langle com_1 ; C_2 \rangle Y$

$$\begin{aligned}
\langle C_1 ; C_2 \rangle X &= \langle C_2 \rangle (\langle C_1 \rangle X) \\
\langle C_1 ; C_2 \rangle Y &= \langle C_2 \rangle (\langle C_1 \rangle Y)
\end{aligned}$$

By induction hypothesis $\langle C_1 \rangle X \subseteq \langle C_1 \rangle Y$, and by induction hypothesis again $\langle C_2 \rangle (\langle C_1 \rangle X) \subseteq \langle C_2 \rangle (\langle C_1 \rangle Y)$ which means $\langle C_1 ; C_2 \rangle X \subseteq \langle C_1 ; C_2 \rangle Y$.

- $\mathsf{C} \equiv \mathsf{C}^*$ therefore we need to show that $\langle \mathsf{C}^* \rangle X \subseteq \langle \mathsf{C}^* \rangle Y$.

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

$$\langle \mathsf{C}^* \rangle Y = \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i Y$$

what we need to prove is that

$$\forall j \in \mathbb{N} \quad \bigcup_{i=0}^{j} \langle \mathsf{C} \rangle^i X \subseteq \bigcup_{i=0}^{j} \langle \mathsf{C} \rangle^i Y$$

we can do this by induction on j:

- $j = 0$ therefore $X \subseteq Y$ which is true by hypothesis.
- Now we need to work on the inductive case $j \Rightarrow j+1$. Notice that it holds that

$$\bigcup_{i=0}^{k+1} \langle \mathsf{C} \rangle^i X = X \cup \bigcup_{i=1}^{k+1} \langle \mathsf{C} \rangle^i X \qquad\qquad \text{by definition}$$

$$= X \cup \langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i X \right) \qquad\qquad \text{by additivity}$$

and also for Y

$$\bigcup_{i=0}^{k+1} \langle \mathsf{C} \rangle^i Y = Y \cup \langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i Y \right)$$

Also notice that

(i) $X \subseteq Y$ by hypothesis;

(ii) $\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i X \subseteq \bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i Y$ by inductive hypothesis;

(iii) $\langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i X \right) \subseteq \langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i Y \right)$ by additivity.

Therefore

$$\bigcup_{i=0}^{k+1} \langle \mathsf{C} \rangle^i X = X \cup \langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i X \right) \subseteq Y \cup \langle \mathsf{C} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{C} \rangle^i Y \right) = \bigcup_{i=0}^{k+1} \langle \mathsf{C} \rangle^i Y$$

$\square$

Since concrete semantics is additive, the Kleene star (C^*) and the fixpoint ($\mathsf{fix}(\mathsf{C})$) have the same concrete semantics $\langle \mathsf{C}^* \rangle = \langle \mathsf{fix}(\mathsf{C}) \rangle$. In order to notice this, let $X \in \mathbb{C}, f = \lambda Y \in \mathbb{C}. (X \cup \langle \mathsf{C} \rangle Y)$ and recall that $f^0 X = X$ and $f^{n+1} X = X \cup \langle \mathsf{C} \rangle (f^n X)$.

$$\begin{aligned} \langle \mathsf{fix}(\mathsf{C}) \rangle X = \mathrm{lfp}(f) &= \bigcup \{ f^n \perp \mid n \in \mathbb{N} \} && \text{by Knaster-Tarsky theorem} \\ &= \bigcup_{i \in \mathbb{N}} \left(X \cup \langle \mathsf{C} \rangle^i X \right) && \text{by definition} \\ &= X \cup \left(X \cup \langle \mathsf{C} \rangle X \right) \cup \left(X \cup \langle \mathsf{C} \rangle X \cup \langle \mathsf{C} \rangle^2 X \right) \cup \ldots \\ &= \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i X \\ &= \langle \mathsf{C}^* \rangle X \end{aligned}$$

This will not be the case for the abstract semantics (cf. example 3.1), 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 2.1 (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 $\langle\mathsf{C}\rangle\{\rho\}$.

2.2.1 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.

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

2.2.2 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*: $\mathsf{State} \triangleq \mathsf{Imp} \times \mathsf{Env}$ tuples of programs and program environments. With states we can define our small step semantics:

Definition 2.3 (Small step semantics)**.** The small step transition relation for the language Imp $\rightarrow: \mathsf{State} \times (\mathsf{State} \cup \mathsf{Env})$ is defined by the following rules:

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

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

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

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

In the following chapters we will usually use the following notation to talk about program execution:

- $\rightarrow^+$ is the transitive closure of the $\rightarrow$ relation;

- $\rightarrow^*$ is the reflexive and transitive closure of the $\rightarrow$ relation.

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

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

where $\rightarrow^*$ *is the reflexive and transitive closure of the* $\rightarrow$ *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

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

- $\mathsf{C} \equiv \mathsf{C}^*$

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

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

$\hfill \square$

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.

2.3   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 2.4 (Transition system). The transition system for the language Imp is

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

where

- $\mathsf{State} \cup \mathsf{Env}$ is the set of configurations in the system;

- Env is the set of terminal states;

- $\rightarrow$ is the small step semantics defined in definition 2.3, which describes the transition relations in the system.

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

Definition 2.5 (Partial termination). Let $\mathsf{C} \in \mathrm{Imp}, \rho \in \mathsf{Env}$. C *partially halts* on ρ when there's at least one derivation sequence of finite length in the transition system $\langle \mathsf{C}, \rho \rangle$ ending up in some state ρ':

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

Dually

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

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

Definition 2.6 (Universal termination). Let $\mathsf{C} \in \mathrm{Imp}, \rho \in \mathsf{Env}$. C *partially loops* on ρ when there's at least one derivation sequence of infinite length in the transition system $\langle \mathsf{C}, \rho \rangle$:

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

Dually

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

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

Example 2.3 shows a program that partially halts, while example 2.2 shows a program that always loops. Notice that the absence of infinite derivation sequences implies that $\langle \mathsf{C}, \rho \rangle$ is finite. Example 2.3 shows a program that partially loops, while example 2.1 shows a program that universally halts.

Example 2.1. Consider the program

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

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

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

according to the expr rule in definition 2.3. Therefore $\langle (\mathtt{x} := 0), \rho \rangle \Downarrow \forall \rho \in \mathsf{Env} \setminus \{\bot\}$.

Example 2.2. Consider the program P
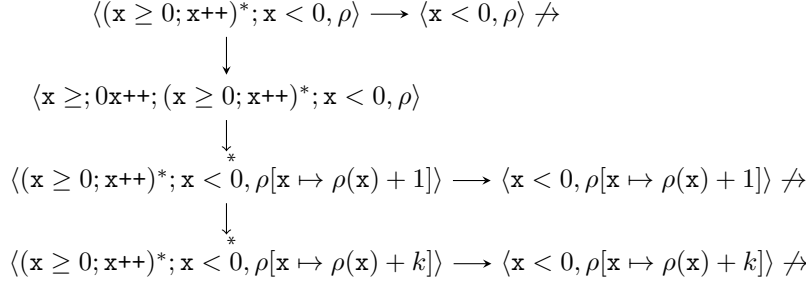
$$(\mathtt{x} \geq 0; \mathtt{x}\mathtt{++})^*; \mathtt{x} < 0$$

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 2.1, where the infinite derivation sequence

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

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

is always present.

$$\langle (x \geq 0; x\text{++})^*; x < 0, \rho \rangle \longrightarrow \langle x < 0, \rho \rangle \nrightarrow$$
$$\downarrow$$
$$\langle x \geq; 0x\text{++}; (x \geq 0; x\text{++})^*; x < 0, \rho \rangle$$
$$\downarrow$$
$$\langle (x \geq 0; x\text{++})^*; x < 0, \rho[x \mapsto \rho(x) + 1] \rangle \xrightarrow{*} \langle x < 0, \rho[x \mapsto \rho(x) + 1] \rangle \nrightarrow$$
$$\downarrow$$
$$\langle (x \geq 0; x\text{++})^*; x < 0, \rho[x \mapsto \rho(x) + k] \rangle \xrightarrow{*} \langle x < 0, \rho[x \mapsto \rho(x) + k] \rangle \nrightarrow$$

Figure 2.1: Transition system of $(x \geq 0; x\text{++})^*; x < 0$

Example 2.3. Consider the program

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

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

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

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

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

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

Lemma 2.3 (Decomposition lemma). *If $\langle C_1; C_2, \rho \rangle \to^k \rho''$, then there exists a state ρ' and a natural number k_1, k_2 s.t. $\langle C_1, \rho \rangle \to^{k_1} \rho'$ and $\langle C_2, \rho' \rangle \to^{k_2} \rho''$, where $k_1 + k_2 = k$*

Corollary 2.1. *If $\langle C_1; C_2, \rho \rangle \to^* \rho''$ then $\exists \rho'$ s.t. $\langle C_1, \rho \rangle \to^* \rho'$ and $\langle C_2, \rho' \rangle \to^* \rho''$.*

Lemma 2.4 (Composition lemma). *If $\langle C_1, \rho \rangle \to^k \rho'$ then $\langle C_1; C_2, \rho \rangle \to^k \langle C_2, \rho' \rangle$*

Corollary 2.2. *If $\langle C_1, \rho \rangle \to^* \rho'$ then $\langle C_1; C_2, \rho \rangle \to^* \langle C_2, \rho' \rangle$.*

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

Definition 2.7 (Reducts). Let Imp^* denotes the set whose elements are statements in Imp. The reduction function $\mathsf{red} : \mathsf{Imp} \to \mathsf{Imp}^*$ is recursively defined by the following clauses:

$$\mathsf{red}(e) \triangleq \{e\}$$
$$\mathsf{red}(C_1 + C_2) \triangleq \{C_1 + C_2\} \cup \mathsf{red}(C_1) \cup \mathsf{red}(C_2)$$
$$\mathsf{red}(C_1; C_2) \triangleq (\mathsf{red}(C_1); C_2) \cup \mathsf{red}(C_2)$$
$$\mathsf{red}(C^*) \triangleq \{C^*\} \cup (\mathsf{red}(C); C^*)$$

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

$$\varnothing; C \triangleq \varnothing$$
$$\{C_1, \dots, C_k\}; C \triangleq \{C_1; C, \dots, C_k; C\}$$

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

2.4 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 arity k:

Definition 2.8 (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:

$$z : \mathbb{N}^k \to \mathbb{N}$$
$$(x_1, \ldots, x_k) \mapsto 0$$

(b) the successor function

$$s : \mathbb{N} \to \mathbb{N}$$
$$x_1 \mapsto x_1 + 1$$

(c) the projection function

$$U_i^k : \mathbb{N}^k \to \mathbb{N}$$
$$(x_1, \ldots, x_k) \mapsto x_i$$

and is closed under

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

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

(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 $(a_1, \ldots, a_k)$ as input for an Imp program. We do this by special input states and variables: we can consider initial states $\rho = [\mathtt{x}_1 \mapsto a_1, \ldots, \mathtt{x}_k \mapsto a_k]$ 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 2.2 (Program output)**.** Let $\mathsf{Env} \ni \rho = [\mathtt{x}_1 \mapsto a_1, \ldots, \mathtt{x}_n \mapsto a_n]$. We say

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

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

$$\exists \mathsf{C} \in \mathrm{Imp} \mid \forall (a_1, \ldots, a_k) \in \mathbb{N}^k \wedge b \in \mathbb{N}$$
$$\langle \mathsf{C}, \rho \rangle \Downarrow b \iff (a_1, \ldots, a_k) \in dom(f) \wedge f(a_1, \ldots, a_k) = b$$

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

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 2.10 (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 2.5 (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:

$$z : \mathbb{N}^k \to \mathbb{N}$$
$$(x_1, \ldots, x_k) \mapsto 0$$

  is Imp-computable:

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

- The successor function

$$s : \mathbb{N} \to \mathbb{N}$$
$$x_1 \mapsto x_1 + 1$$

  is Imp-computable:

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

- The projection function

$$U_i^k : \mathbb{N}^k \to \mathbb{N}$$
$$(x_1, \ldots, x_k) \mapsto x_i$$

  is Imp-computable:

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

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

Lemma 2.6. *let* $f : \mathbb{N}^k \to \mathbb{N}$, $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ *and consider the composition*

$$h : \mathbb{N}^k \to \mathbb{N}$$
$$\vec{x} \mapsto f(g_1(\vec{x}), \ldots, g_k(\vec{x}))$$

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

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

Which is exactly h. Therefore Imp is closed under generalized composition. $\square$

Lemma 2.7. *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. $\square$

Lemma 2.8. *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*

$$
h(\vec{x}) = \mu y.f(\vec{x}, y) = \begin{cases} least \ z \ \ s.t. & \begin{cases} f(\vec{x}, z) = 0 \\ f(\vec{x}, z) \downarrow \quad f(\vec{x}, z') \neq 0 \quad \forall z < z' \end{cases} \\ \uparrow & otherwise \end{cases} \tag{2.1}
$$

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. $\square$

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. $\square$

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

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

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

Corollary 2.3. $\langle \mathsf{C}, \rho \rangle \Uparrow$ *(i.e., $\langle \mathsf{C} \rangle X = \varnothing$) is undecidable.*

Proof. The set of functions $f \in \mathbb{N}^k \xrightarrow{r} \mathbb{N}$ 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] $\square$

Corollary 2.4. $\langle \mathsf{C}, \rho \rangle \Downarrow$ *is undecidable.*

Proof. The set of functions $f \in \mathbb{N}^k \xrightarrow{r} \mathbb{N}$ 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]; $\square$

2.5   Deciding invariant finiteness

In this section we argue that even the finiteness of the semantics of some program on some initial states is undecidable. We show that by knowing whether $\langle \mathsf{C} \rangle X$ is finite for some program $\mathsf{C} \in \text{Imp}$ and some initial states $X \in \mathbb{C}$, we can decide whether $\langle \mathsf{C}, \rho \rangle \Downarrow$ for all $\rho \in X$. The first step is showing that if we have a program where the * operator does not appear, then the program can only produce a finite amount of finite derivation sequences.

Lemma 2.9. *If $\mathsf{D} \in Imp_{\neq \star}$, and $X \in 2^{env}$ is finite, then*

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

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

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

   where by $|\langle \mathsf{D}, \rho \rangle|$ we mean the number of derivation sequences starting from $\langle \mathsf{D}, \rho \rangle$.

Proof. By induction on the program D:

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

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

(ii). by expr rule $\forall \rho \in X$ either $\langle e, \rho \rangle \to (\!| e |\!) \rho$ or $\langle e, \rho \rangle \nrightarrow$. In both cases there are no infinite derivation sequences, and therefore $\langle e, \rho \rangle \Downarrow$;

(iii). Notice that $\forall \rho \in X$ either by the expr rule $\langle e, \rho \rangle \to (\!| e |\!) \rho$ or $\langle e, \rho \rangle \nrightarrow$ therefore

$$|\langle e, \rho \rangle| \leqslant |X| < \infty$$

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 $\forall \rho \in X \ \langle \mathsf{D}_1, \rho \rangle \Downarrow$ and $\langle \mathsf{D}_2, \rho \rangle \Downarrow$. 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 $\langle \mathsf{C}_1 + \mathsf{C}_2, \rho \rangle \Downarrow$.

(iii). For the latter argument, since both $\langle D_1, \rho \rangle$ and $\langle D_2, \rho \rangle$ are finite and composed of finite derivation sequences $|\langle (D_1 + D_2), \rho \rangle| < \infty$.

2. $D \equiv D_1; D_2$, therefore

(i). $\langle D_1; D_2 \rangle X = \langle D_2 \rangle (\langle D_1 \rangle X)$. By inductive hypothesis $\langle D_1 \rangle X = Y$. By inductive hypothesis again $\langle D_2 \rangle Y$ is finite;

(ii). by inductive hypothesis both $\forall \rho \in X \ \langle D_1, \rho \rangle \Downarrow$ and $\forall \rho' \in Y \ \langle D_2, \rho' \rangle \Downarrow$, therefore by composition lemma $\langle D_1; D_2, \rho \rangle \Downarrow$

(iii). by inductive hypothesis both $|\langle D_1, \rho \rangle| < \infty$ and $|\langle D_2, \rho' \rangle| < \infty \ \forall \rho \in X, \rho' \in \langle D_1 \rangle X$. For all derivation sequences starting from $\langle D_1, \rho \rangle$ where

$$\langle D_1, \rho \rangle \to^* \rho'$$

with $\rho' \in \langle D_1 \rangle X$ we can apply the composition lemma and state that

$$\langle D_1; D_2, \rho \rangle \to^* \langle D_2, \rho' \rangle \quad \forall \rho \in X$$

from there we can notice that since $|\langle D_2, \rho' \rangle| < \infty$ then $|\langle D_1; D_2, \rho' \rangle| < \infty$

$\square$

Lemma 2.10. *Given* $D \in Imp_{\neq \star}$, *and* $\rho \in \mathsf{Env}$, *the predicate* "$\langle D^* \rangle \rho$ *is finite*" *is undecidable.*

Proof. We work by contradiction, showing that if we know whether $\langle C \rangle \rho$ is finite or infinite we can decide $\langle C, \rho \rangle \Downarrow$.

- Suppose we can decide that $\langle D^* \rangle \rho$ is infinite, then we make the following observation:

Observation 2.2.

$$\forall k \in \mathbb{N} \quad \langle D \rangle^{k+1} \rho \nsubseteq \bigcup_{i=0}^{k} \langle D \rangle^i \rho$$

The latter observation follows from the following lemma:

Lemma 2.11. *Let* $D \in Imp_{\neq \star}$ $\rho \in \mathsf{Env}$. *If*

$$\langle D \rangle^{k+1} \rho \subseteq \cup_{i=0}^{k} \langle D \rangle^i \rho \quad \text{for some } k \in \mathbb{N} \tag{2.2}$$

then

$$\forall j \in \mathbb{N} \quad \langle D \rangle^{k+1+j} \rho \subseteq \cup_{i=0}^{k} \langle D \rangle^i \rho \tag{2.3}$$

and therefore $\langle D^* \rangle \rho \subseteq \cup_{i=0}^{k} \langle D \rangle^i \rho$

Proof. We can show (2.3) by induction on j:

- if $j = 0$ then we want to show that $\langle D \rangle^{k+1} \rho \subseteq \cup_{i=0}^{k} \langle D \rangle^i \rho$, which is true by hypothesis (2.2);

- In the inductive case we have to show that if the statement holds for j, it also holds for $j + 1$. We know that

$$\bigcup_{i=0}^{k} \langle D \rangle^i \rho = \bigcup_{i=0}^{k+1} \langle D \rangle^i \rho \qquad \text{since by (2.2) } \langle D \rangle^{k+1} \rho \subseteq \cup_{i=0}^{k} \langle D \rangle^i \rho$$

$$= \rho \cup \bigcup_{i=1}^{k+1} \langle D \rangle^i \rho$$

$$= \rho \cup \langle D \rangle \left(\bigcup_{i=0}^{k} \langle D \rangle^i \rho \right) \qquad \text{by additivity}$$

By inductive hypothesis

$$\langle \mathsf{D} \rangle^{k+1+j} \rho \subseteq \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho$$

so, by monotonicity of $\langle \mathsf{D} \rangle$

$$\langle \mathsf{D} \rangle \left(\langle \mathsf{D} \rangle^{k+1+j} \rho \right) \subseteq \langle \mathsf{D} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho \right)$$

and therefore

$$\langle \mathsf{D} \rangle^{(k+1)+(j+1)} \rho \subseteq \left(\bigcup_{i=1}^{k+1} \langle \mathsf{D} \rangle^{i} \rho \right) \subseteq \rho \cup \left(\bigcup_{i=1}^{k+1} \langle \mathsf{D} \rangle^{i} \rho \right) = \bigcup_{i=0}^{k+1} \langle \mathsf{D} \rangle^{i} \rho = \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho$$

$$\square$$

therefore $\forall k \in \mathbb{N} \ \langle \mathsf{D} \rangle^{k+1} \rho \not\subseteq \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho$, otherwise $\langle \mathsf{D}^{*} \rangle \rho \subseteq \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle \rho$ which is impossible since the right term is a finite quantity. With this observation we build the tree $\langle \mathsf{Env}, \rightarrow^{D} \rangle$, where $\rightarrow^{D} \subseteq \mathsf{Env} \times \mathsf{Env}$ and if $\rho' \rightarrow^{D} \rho''$ then $\langle \mathsf{D}, \rho' \rangle \rightarrow^{*} \rho''$. We define by the following rule the levels of the tree:

$$Y_0 = \{\rho\}$$

$$Y_{k+1} = \left(\langle \mathsf{D} \rangle^{k+1} \rho \right) \setminus \left(\bigcup_{i=0}^{k} \langle \mathsf{D} \rangle \rho \right)$$

Where Y_0 is the singleton set containing the root ρ and the k-th level is made of the environments in the Y_k set. Figure 2.2 shows a tree of $\rightarrow^{D}$ relations and visualizes the levels Y_k. We can therefore make the following observations:

(i) The tree is rooted in $\rho \in Y_0$. In fact $\forall \rho' \in Y_1 \ \rho \rightarrow^{D} \rho'$ by definition and $\forall \rho''' \in Y_{k+1} \exists \rho'' \in Y_k \mid \rho'' \rightarrow^{D} \rho'''$;

(ii) because of observation 2.2 each level Y_k is non empty and finite, therefore there is an infinite quantity of levels, where each node has a finite quantity of children;

(iii) for all $\rho', \rho'' \in \mathsf{Env}$ s.t. $\rho' \rightarrow^{D} \rho''$ since $\langle \mathsf{D}, \rho' \rangle \rightarrow^{*} \rho''$ we can apply the corollary (2.2) of the composition lemma (2.4) and state that $\langle \mathsf{D}; \mathsf{D}^{*}, \rho' \rangle \rightarrow^{*} \langle \mathsf{D}^{*}, \rho'' \rangle$ and because of the star rule $\langle \mathsf{D}^{*}, \rho' \rangle \rightarrow \langle \mathsf{D}; \mathsf{D}^{*}, \rho' \rangle$ We can therefore say that

$$\langle \mathsf{D}^{*}, \rho' \rangle \rightarrow^{*} \langle \mathsf{D}^{*}, \rho'' \rangle$$

what is left to do is show that there's a derivation sequence from $\langle \mathsf{D}^{*}, \rho \rangle$ of infinite length. To do so we need König's Lemma from [Kön26]:

Lemma 2.12 (König's Lemma). *Let T be a rooted tree with an infinite number of nodes, each with a finite number of children. Then T has a branch of infinite length.*

We can therefore use König's Lemma (2.12) and deduce that there exists an infinite derivation sequence from ρ of $\rightarrow^{D}$ relations

$$\rho \rightarrow^{D} \rho' \rightarrow^{D} \rho'' \rightarrow^{D} \dots$$

Where each element belongs to a different level Y_k, and therefore is different from every other environment appearing in the sequence. Because of observation (iii) there exists a derivation sequence

$$\langle \mathsf{D}^{*}, \rho \rangle \rightarrow^{*} \langle \mathsf{D}^{*}, \rho' \rangle \rightarrow^{*} \langle \mathsf{D}^{*}, \rho'' \rangle \rightarrow^{*} \dots$$

which is infinite, which means $\langle \mathsf{D}^{*}, \rho \rangle \uparrow$ and therefore $\langle \mathsf{D}^{*}, \rho \rangle \Downarrow$ is false.
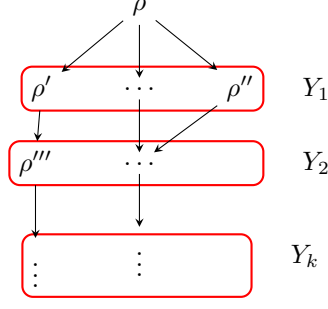
Figure 2.2: Example of $\to^D$ relations between elements of Env.

- Suppose we can decide $\langle \mathsf{D}^* \rangle \rho$ is finite, then we can reduce total termination to the presence of some cycle in one of the derivation sequences starting from $\langle \mathsf{D}^*, \rho \rangle$. The statement we want to prove is the following:

  if $\langle \mathsf{D}^* \rangle \rho$ is finite, then $\langle \mathsf{D}^*, \rho \rangle \Downarrow \iff$ no derivation sequence starting from $\langle \mathsf{D}^*, \rho \rangle$ has cycles

  ($\Rightarrow$) In this case we want to prove that if $\langle \mathsf{D}^* \rangle$ is finite and $\langle \mathsf{D}, \rho \rangle \Downarrow$ then there are no cycles in any derivation sequence starting from $\langle \mathsf{D}, \rho \rangle$. To do so we work by contradiction. Suppose there is some derivation sequence starting from $\langle \mathsf{D}^*, \rho \rangle$ with some cycle

  $$\langle \mathsf{D}^*, \rho \rangle \to^* \langle \mathsf{D}^*, \rho' \rangle \to^+ \langle \mathsf{D}^*, \rho' \rangle \to^* \rho''$$

  with $\rho'' \neq \rho, \rho'$, then we can notice that also the infinite derivation sequence

  $$\langle \mathsf{D}^*, \rho \rangle \to^* \langle \mathsf{D}^*, \rho' \rangle \to^+ \langle \mathsf{D}^*, \rho' \rangle \to^+ \langle \mathsf{D}^*, \rho' \rangle \to^+ \ldots$$

  is part of the transition system for $\langle \mathsf{D}, \rho \rangle$, and therefore $\langle \mathsf{D}^*, \rho \rangle \Downarrow$ is false which is absurd.

  ($\Leftarrow$) In this case we want to prove that if $\langle \mathsf{D}^* \rangle \rho$ is finite and there are no cycles in any derivation sequence starting from $\langle \mathsf{D}, \rho \rangle$ then $\langle \mathsf{D}, \rho \rangle \Downarrow$. We work again by contradiction. Suppose that we have an infinite derivation sequence starting from $\langle \mathsf{D}^*, \rho \rangle$. It must be that $\forall i, j \in \mathbb{N} \ i \neq j, \ \rho_i \neq \rho_j$ with $\rho_0 = \rho$, otherwise there would be a cycle, which is not the case by hypothesis. Therefore the derivation sequence would have the shape

  $$\langle \mathsf{D}^*, \rho \rangle \to^* \langle \mathsf{D}^*, \rho_1 \rangle \to^* \langle \mathsf{D}^*, \rho_2 \rangle \to^* \langle \mathsf{D}^*, \rho_3 \rangle \to^* \ldots$$

  We can notice that for all $\rho' \in \{\rho, \rho_1, \ldots\}$ and for the $\mathsf{star}_{\mathsf{fix}}$ rule, $\langle \mathsf{D}^*, \rho' \rangle \to \rho'$ and therefore $\rho' \in \langle \mathsf{D}^* \rangle \rho$. This would mean that $\langle \mathsf{D}^* \rangle \rho$ is infinite, which is absurd.

  $\square$

Chapter 3

Intervals

3.1 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 3.1 (Integer intervals)**.** We call

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

set of integer intervals.

Definition 3.2 (Concretization map)**.** We define the *concretization map* $\gamma : Int \to 2^{\mathbb{Z}}$ as

$$\gamma([a, b]) \triangleq \{x \in \mathbb{Z} \mid a \leqslant x \leqslant b\}$$
$$\gamma(\perp) \triangleq \varnothing$$

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

Definition 3.3 (Abstract integer domain)**.** Let $Int_* \triangleq Int \setminus \{\perp^\sharp\}$. The abstract domain $\mathbb{A}$ for program analysis is the variable-wise lifting of Int:

$$\mathbb{A} \triangleq (Var \to Int_*) \cup \{\perp^\sharp\}$$

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

Definition 3.4 (Interval concretization)**.** We define the *concretization map* for the abstract domain $\mathbb{A}$ $\gamma_{Int} : \mathbb{A} \to 2^{\mathsf{Env}}$ as

$$\gamma_{Int}(\perp) \triangleq \varnothing$$
$$\forall \eta \neq \perp \quad \gamma_{Int}(\eta) \triangleq \{\rho \in \mathsf{Env} \mid \forall x \in Var \; \rho(x) \in \gamma(\eta(x))\}$$

Observation 3.1. If we consider the ordering $\sqsubseteq$ on $\mathbb{A}$ s.t.

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

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

Definition 3.5 (Interval abstraction)**.** We define the *abstraction map* of some numerical set $X \subseteq \mathbb{Z}$ into the abstract domain $\mathbb{A}$: $\alpha_{Int} : 2^{\mathbb{Z}} \to \mathbb{A}$ as

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

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

$$[a,b] \sqcup [c,d] \triangleq [\min(a,c), \max(b,d)]$$

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

And sound abstract arithmetical operations:

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

Definition 3.7 (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:

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

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

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

Definition 3.8 (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 $\pm\infty + c = \pm\infty - c = \pm\infty$). $\qquad\square$

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 $\llbracket \cdot \rrbracket : \text{Imp} \to \mathbb{A} \to \mathbb{A}$. For all $\eta : Var \to Int_*$,

$$\llbracket \mathsf{x} \in S \rrbracket \eta \triangleq \begin{cases} \eta[\mathsf{x} \mapsto \eta(\mathsf{x}) \sqcap \alpha_{Int}(S)] & \text{if } \eta(\mathsf{x}) \sqcap \alpha_{Int}(S) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$\llbracket \mathsf{x} \in [a, b] \rrbracket \eta \triangleq \begin{cases} \eta[\mathsf{x} \mapsto \eta(\mathsf{x}) \sqcap [a, b]] & \text{if } \eta(\mathsf{x}) \sqcap [a, b] \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$\llbracket \mathsf{x} \leqslant k \rrbracket \eta \triangleq \begin{cases} \eta[\mathsf{x} \mapsto \eta(\mathsf{x}) \downarrow k] & \text{if } \eta(\mathsf{x}) \downarrow k \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$\llbracket \mathsf{x} > k \rrbracket \eta \triangleq \begin{cases} \eta[\mathsf{x} \mapsto \eta(\mathsf{x}) \uparrow k + 1] & \text{if } \eta(\mathsf{x}) \downarrow k \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$\llbracket \mathsf{true} \rrbracket \eta \triangleq \eta$$

$$\llbracket \mathsf{false} \rrbracket \eta \triangleq \bot$$

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

$$\llbracket \mathsf{x} := \mathsf{y} + k \rrbracket \eta \triangleq \eta[\mathsf{x} \mapsto \eta(\mathsf{y}) + k]$$

$$\llbracket \mathsf{x} := \mathsf{x} - k \rrbracket \eta \triangleq \eta[\mathsf{x} \mapsto \eta(\mathsf{y}) - k]$$

$$\llbracket \mathsf{C}_1 + \mathsf{C}_2 \rrbracket \eta \triangleq \llbracket \mathsf{C}_1 \rrbracket \eta \sqcup \llbracket \mathsf{C}_2 \rrbracket \eta$$

$$\llbracket \mathsf{C}_1 ; \mathsf{C}_2 \rrbracket \eta \triangleq \llbracket \mathsf{C}_2 \rrbracket (\llbracket \mathsf{C}_1 \rrbracket \eta)$$

$$\llbracket \mathsf{C}^* \rrbracket \eta \triangleq \bigsqcup_{i \in \mathbb{N}} \llbracket \mathsf{C} \rrbracket^i (\eta)$$

$$\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \triangleq \mathrm{lfp} \lambda \mu. (\eta \sqcup \llbracket \mathsf{C} \rrbracket \mu)$$

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

Lemma 3.1. *for all* $\mathsf{C} \in Imp,$

$$\llbracket \mathsf{C} \rrbracket : \mathbb{A} \to \mathbb{A}$$

is monotone.

Proof. What we have to proof is that given $\eta, \vartheta \in \mathbb{A}$, with $\eta \sqsubseteq \vartheta$ then $\forall \mathsf{C} \in Imp$ $\llbracket \mathsf{C} \rrbracket \eta \sqsubseteq \llbracket \mathsf{C} \rrbracket \vartheta$. We will work by induction on C:

Base cases:

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

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

$$\llbracket \mathsf{x} \in S \rrbracket \eta = \eta[x \mapsto \eta(x) \sqcap Int(S)]$$
$$\llbracket \mathsf{x} \in S \rrbracket \vartheta = \vartheta[x \mapsto \vartheta(x) \sqcap Int(S)]$$

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

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

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

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

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

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

$$(3.1)$$

  Notice that

$$
[\![\mathrm{x} := k]\!]\eta = \eta[\mathrm{x} \mapsto [k,k]]
$$

$$
[\![\mathrm{x} := k]\!]\vartheta = \vartheta[\mathrm{x} \mapsto [k,k]]
$$

  because of equation 3.1 in this case we know that $\forall \mathrm{y} \in Var,\ \mathrm{y} \neq \mathrm{x}\ \rho(\mathrm{y}) \in \gamma(\eta(\mathrm{y})) \Rightarrow \rho(\mathrm{y}) \in \gamma(\vartheta(\mathrm{y}))$. For x it holds that $\rho(\mathrm{x}) \in \gamma([k,k]) \Rightarrow \rho(\mathrm{x}) \in \gamma([k,k])$ and therefore

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

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

Recursive cases:

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

$$
\begin{aligned}
[\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta &= [\![\mathsf{C}_1]\!]\eta \sqcup [\![\mathsf{C}_2]\!]\eta \\
&\sqsubseteq [\![\mathsf{C}_1]\!]\vartheta \sqcup [\![\mathsf{C}_2]\!]\vartheta \qquad \text{by inductive hp.} \\
&= [\![\mathsf{C}_1 + \mathsf{C}_2]\!]\vartheta
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\mathsf{C}_1; \mathsf{C}_2]\!]\eta &= [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\eta) \\
\alpha &= [\![\mathsf{C}_1]\!]\eta \sqsubseteq [\![\mathsf{C}_1]\!]\vartheta = \beta \qquad \text{by inductive hp.} \\
&[\![\mathsf{C}_2]\!]\alpha \sqsubseteq [\![\mathsf{C}_2]\!]\beta \qquad\qquad\quad \text{by inductive hp.} \\
&[\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\eta) \sqsubseteq [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\vartheta) \qquad \text{by substitution}
\end{aligned}
$$

- C^*. What we first want to prove is that

$$
\forall i \in \mathbb{N}.[\![\mathsf{C}]\!]^i \eta \sqsubseteq [\![\mathsf{C}]\!]^i \vartheta
$$

  To do so we can work by induction on i:

  – with $i = 0$ we have to prove that

$$
[\![\mathsf{C}]\!]^0 \eta \sqsubseteq [\![\mathsf{C}]\!]^0 \vartheta
$$

  since $[\![\mathsf{C}]\!]^0 = id$ the statement is true by hypothesis.
  – the inductive step is to prove the statement for $i + 1$ with the hypothesis that it holds for i.

$$
[\![\mathsf{C}]\!]^{i+1} \eta = [\![\mathsf{C}]\!]\left([\![\mathsf{C}]\!]^i \eta\right)
$$

$$
[\![\mathsf{C}]\!]^{i+1} \vartheta = [\![\mathsf{C}]\!]\left([\![\mathsf{C}]\!]^i \vartheta\right)
$$

  By inductive hypothesis on i $[\![\mathsf{C}]\!]^i \eta \sqsubseteq [\![\mathsf{C}]\!]^i \vartheta$, while by inductive hypothesis on C

$$
[\![\mathsf{C}]\!]\left([\![\mathsf{C}]\!]^i \eta\right) \sqsubseteq [\![\mathsf{C}]\!]\left([\![\mathsf{C}]\!]^i \vartheta\right)
$$

therefore

$$\llbracket C \rrbracket^{i+1}\eta = \llbracket C \rrbracket \left(\llbracket C \rrbracket^i \eta \right) \sqsubseteq \llbracket C \rrbracket \left(\llbracket C \rrbracket^i \vartheta \right) = \llbracket C \rrbracket^{i+1}\vartheta$$

finally we can notice that $\forall k \in \mathbb{N}$

$$\bigsqcup_{i=0}^{k} \llbracket C \rrbracket^i \eta \sqsubseteq \bigsqcup_{i=0}^{k} \llbracket C \rrbracket^i \vartheta$$

and therefore

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

$\square$

Example 3.1. 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:

$$\text{Kleene:} \quad \sqcup \{[3, 4], [0, 1], [0, 0], [0, 0], \ldots\} = [0, 4]$$

$$\text{Fix:} \quad \sqcup \{\bot, [3, 4], [0, 4], [0, 5], [0, 5], \ldots\} = [0, 5]$$

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 3.2 (fix(C) **is syntactic sugar**)**.** *For all η, $\llbracket \text{fix}(C) \rrbracket \eta = \llbracket (\text{true} + C)^* \rrbracket \eta$.*

Proof. Let us first show by induction that

$$\forall i \geq 0. \, (\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot = (\text{true} \sqcup \llbracket C \rrbracket)^i \eta \qquad (\sharp)$$

$i = 0$: $(\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^1 \bot = \eta \sqcup \bot \sqcup \llbracket C \rrbracket \bot = \eta = (\text{true} \sqcup \llbracket C \rrbracket)^0 \eta$.
$i + 1$:

$$
\begin{aligned}
(\text{true} \sqcup \llbracket C \rrbracket)^{i+1}\eta &= \\
(\text{true} \sqcup \llbracket C \rrbracket)((\text{true} \sqcup \llbracket C \rrbracket)^i \eta) &= \\
((\text{true} \sqcup \llbracket C \rrbracket)^i \eta) \sqcup \llbracket C \rrbracket ((\text{true} \sqcup \llbracket C \rrbracket)^i \eta) &= \qquad \text{By induction}\\
(\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot \sqcup \llbracket C \rrbracket ((\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot) &= \qquad \text{As } \eta \sqsubseteq (\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot\\
\eta \sqcup (\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot \sqcup \llbracket C \rrbracket ((\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot) &= \\
(\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)((\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+1}\bot) &= \\
(\eta \sqcup \text{true} \sqcup \llbracket C \rrbracket)^{i+2}\bot
\end{aligned}
$$

Let us also show that:

$$\text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu) = \text{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup \llbracket C \rrbracket \mu) \qquad (\diamond)$$

Observe that $\text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu) = \eta \sqcup \llbracket C \rrbracket (\text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu))$, so that we have that:

$$\eta \sqcup \text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu) \sqcup \llbracket C \rrbracket (\text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu)) \sqsubseteq \text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu)$$

As a consequence, $\text{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup \llbracket C \rrbracket \mu) \sqsubseteq \text{lfp}\lambda\mu.(\eta \sqcup \llbracket C \rrbracket \mu)$ holds. The reverse inequality follows because, for all μ, $\eta \sqcup \llbracket C \rrbracket \mu \sqsubseteq \eta \sqcup \mu \sqcup \llbracket C \rrbracket \mu$.

Then, we have that:

$$\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta =$$
$$\mathrm{lfp}\lambda\mu.(\eta \sqcup \llbracket \mathsf{C} \rrbracket \mu) = \qquad\qquad\qquad \text{By } (\diamond)$$
$$\mathrm{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup \llbracket \mathsf{C} \rrbracket \mu) = \qquad\qquad \text{By Knaster-Tarski Theorem}$$
$$\bigsqcup_{i\in\mathbb{N}}(\eta \sqcup \mathsf{true} \sqcup \llbracket \mathsf{C} \rrbracket)^i \bot =$$
$$\bot \sqcup \bigsqcup_{i\in\mathbb{N}}(\eta \sqcup \mathsf{true} \sqcup \llbracket \mathsf{C} \rrbracket)^{i+1} \bot = \qquad\qquad\qquad \text{By } (3.3)$$
$$\bigsqcup_{i\in\mathbb{N}}(\mathsf{true} \sqcup \llbracket \mathsf{C} \rrbracket)^i \eta =$$
$$\llbracket (\mathsf{true} + \mathsf{C})^* \rrbracket \eta.$$

$\square$

Theorem 3.1 (Correctness). *For all* $\mathsf{C} \in Imp$ *and* $\eta \in \mathbb{A}$, $\langle\mathsf{C}\rangle\gamma(\eta) \subseteq \gamma(\llbracket\mathsf{C}\rrbracket\eta)$ *holds.*

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

Base cases:

- $\mathsf{C} \equiv \mathrm{x} \in S$:

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

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

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

- $\mathsf{C} \equiv \mathrm{x} \in [a,b], \mathrm{x} \leqslant k, \mathrm{x} > k$: 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

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

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

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

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

Inductive cases:

- $C \equiv C_1 + C_2$, therefore

$$\langle C_1 + C_2 \rangle \gamma_{Int}(\eta) = \langle C_1 \rangle \gamma_{Int}(\eta) \cup \langle C_2 \rangle \gamma_{Int}(\eta)$$

  and

$$\gamma_{Int}(\llbracket C_1 + C_2 \rrbracket \eta) = \gamma_{Int}(\llbracket C_1 \rrbracket \eta \sqcup \llbracket C_2 \rrbracket \eta) = \gamma_{Int}(\llbracket C_1 \rrbracket \eta) \cup \gamma_{Int}(\llbracket C_2 \rrbracket \eta).$$

  By inductive hypothesis both $\langle C_1 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket C_1 \rrbracket \eta)$ and $\langle C_2 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket C_2 \rrbracket \eta)$, therefore

$$\langle C_1 + C_2 \rangle \gamma_{Int}(\eta) \subseteq \gamma_{Int}(\llbracket C_1 + C_2 \rrbracket \eta);$$

- $C \equiv C_1 ; C_2$, therefore $\langle C_1 ; C_2 \rangle \gamma_{Int}(\eta) = \langle C_2 \rangle (\langle C_1 \rangle \gamma_{Int}(\eta))$, while

$$\gamma_{Int}(\llbracket C_1 ; C_2 \rrbracket \eta) = \gamma_{Int}(\llbracket C_2 \rrbracket (\llbracket C_1 \rrbracket \eta)).$$

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

$\square$

Remark 3.1. 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, $\llbracket C \rrbracket \eta$ might be infinite on some variable when C includes a either Kleene or fix iteration which does not converge in finitely many steps.

3.2 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 2.5.

Problem 3.1 (Termination of interval analysis). Given $C \in \text{Imp}$, $\eta \in \mathbb{A}$, decide: $\llbracket C \rrbracket \eta =^? \top$

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

$$(\mathtt{x} \in S)^{\flat} \triangleq \begin{cases} \min(S) & \text{if } \max(S) = \infty \\ \max(S) & \text{if } \max(S) \in \mathbb{N} \end{cases}$$

$$(\mathtt{x} \in [a, b])^{\flat} \triangleq \begin{cases} a & \text{if } b = \infty \\ b & \text{if } b \in \mathbb{N} \end{cases}$$

$$(\mathtt{x} \leqslant k)^{\flat} \triangleq k$$

$$(\mathtt{x} > k)^{\flat} \triangleq k$$

$$(\mathtt{true})^{\flat} \triangleq 0$$

$$(\mathtt{false})^{\flat} \triangleq 0$$

$$(\mathtt{x} := k)^{\flat} \triangleq k$$

$$(\mathtt{x} := \mathtt{y} + k)^{\flat} \triangleq k$$

$$(\mathtt{x} := \mathtt{y} - k)^{\flat} \triangleq k$$

$$(C_1 + C_2)^{\flat} \triangleq (C_1)^{\flat} + (C_2)^{\flat}$$

$$(C_1; C_2)^{\flat} \triangleq (C_1)^{\flat} + (C_2)^{\flat}$$

$$(C^{*})^{\flat} \triangleq (|vars(C)| + 1)(C)^{\flat}$$

where $vars(C)$ denotes the set of variables occurring in C.

Definition 3.10 (**Bound Environment**). A bound environment (benv for short) is a total function $b : Var \to \mathbb{N}$. We define $\mathsf{bEnv} \triangleq \{b \mid b : Var \to \mathbb{N}\}$. Each command $C \in \text{Imp}$ induces a benv transformer $[C]^{\flat} : \mathsf{bEnv} \to \mathsf{bEnv}$, which is defined inductively as follows:

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

$$[\mathtt{x} := k]^{\flat} b \triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + k]$$

$$[\mathtt{x} := \mathtt{y} + k]^{\flat} b \triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + b(\mathtt{y}) + k]$$

$$[\mathtt{x} := \mathtt{y} - k]^{\flat} b \triangleq b[\mathtt{x} \mapsto b(\mathtt{x}) + b(\mathtt{y}) - k]$$

$$[C_1 + C_2]^{\flat} b \triangleq \lambda \mathtt{x}.([C_1]^{\flat} b)(\mathtt{x}) + ([C_2]^{\flat} b)(\mathtt{x})$$

$$[C_1; C_2]^{\flat} b \triangleq \lambda \mathtt{x}.([C_1]^{\flat} b)(\mathtt{x}) + ([C_2]^{\flat} b)(\mathtt{x})$$

$$[C^{*}]^{\flat} b \triangleq \lambda \mathtt{x}.(|vars(C)| + 1)([C]^{\flat} b)(\mathtt{x})$$

where $vars(C)$ denotes the set of variables occurring in C.

Lemma 3.3. *For all $C \in Imp$, $(C)^{\flat} = \sum_{\mathtt{x} \in vars(C)}([C]^{\flat} b_0)(\mathtt{x})$, with $b_0 \triangleq \lambda x.0$.*

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

Base cases:

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

$$(\mathtt{x} \in S)^{\flat} = \begin{cases} \min(S) & \text{if } \max(S) = \infty \\ \max(S) & \text{otherwise} \end{cases}$$

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

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

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

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

$(\mathtt{x} := k)$: 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:

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

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

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

(C^*)

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

$\square$

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

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

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 3.4 (**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

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

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. $\square$

Lemma 3.5. *Let $\mathsf{C} \in Imp$ and $\mathsf{y} \in Var$.*
For all $\eta \in \mathbb{A}$ and $\mathsf{y} \in Var$, 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 $\mathsf{z} \in Var$ 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 $\mathtt{y} \neq \mathtt{x}$, then for all $\eta' \in \mathbb{A}$ such that $\eta \sqsubseteq \eta'$ it holds $\bot \neq [\![\mathtt{x} \in S]\!]\eta' = \eta'[\mathtt{x} \mapsto \eta(\mathtt{x}) \sqcap Int(S)]$ and thus
$$\max([\![\mathtt{x} \in S]\!]\eta'\mathtt{y}) = \max(\eta'\mathtt{y}) = \max(\eta'\mathtt{y}) + 0$$
hence the thesis follows with $\mathtt{z} = \mathtt{y}$ and $h = 0$.

- If $\mathtt{y} = \mathtt{x}$ then $\eta(\mathtt{x}) \in Int_*$ and
$$\max([\![\mathtt{x} \in S]\!]\eta\mathtt{y}) = \max(\eta(\mathtt{x}) \sqcap Int(S))$$

  Note that it cannot be $\max(S) \in \mathbb{N}$. Otherwise, by Definition 3.9, $\max(\eta(\mathtt{x}) \sqcap Int(S)) \leqslant \max(S) = (\mathtt{x} \in S)^{\mathsf{b}}$, violating the assumption $\max([\![\mathtt{x} \in S]\!]\eta\mathtt{y}) > (\mathtt{x} \in S)^{\mathsf{b}}$. Hence, $\max(S) = \infty$ must hold and therefore $\max(\eta(\mathtt{x}) \sqcap Int(S)) = \max(\eta(\mathtt{x})) = \max(\eta(\mathtt{x})) + 0$. It is immediate to check that the same holds for all $\eta' \sqsupseteq \eta$, i.e.,
$$\max(\eta'(\mathtt{x}) \sqcap Int(S)) = \max(\eta'(\mathtt{x})) = \max(\eta'(\mathtt{x})) + 0$$
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 $\max([\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta) > (\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$.

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

Moreover

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

Thus $\max([\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta\mathtt{y}) = \max([\![\mathsf{C}_i]\!]\eta\mathtt{y})$ for some $i \in \{1, 2\}$. We can assume, without loss of generality, that the maximum is realized by the first component, i.e., $\max([\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta\mathtt{y}) = \max([\![\mathsf{C}_1]\!]\eta\mathtt{y})$. 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 $\max([\![\mathsf{C}_1]\!]\eta\mathtt{y}) = \max(\eta\mathtt{z}) + h$ and for all $\eta' \in \mathbb{A}$, $\eta \sqsubseteq \eta'$,

$$\max([\![\mathsf{C}_1]\!]\eta'\mathtt{y}) \geq \max(\eta'\mathtt{z}) + h$$

Therefore

$$\max([\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta\mathtt{y}) = \max([\![\mathsf{C}_1]\!]\eta\mathtt{y}) = \max(\eta\mathtt{z}) + h$$

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

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

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 $\max([\![\mathsf{C}_1; \mathsf{C}_2]\!]\eta) > (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} = (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$.

Recall that $[\![\mathsf{C}_1; \mathsf{C}_2]\!]\eta = [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\eta)$. If we define

$$[\![\mathsf{C}_1]\!]\eta = \eta_1$$

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 $\max([\![\mathsf{C}_2]\!]\eta_1\mathtt{y}) = \max(\eta_1\mathtt{w}) + h_2$ and for all $\eta_1' \in \mathbb{A}$ with $\eta_1 \sqsubseteq \eta_1'$

$$\max([\![\mathsf{C}_2]\!]\eta_1'\mathtt{y}) \geq \max(\eta_1'\mathtt{w}) + h_2 \tag{$\dagger$}$$

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

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

violating the hypotheses. Moreover, $[\![\mathsf{C}_1]\!]\eta\mathtt{w} \neq \infty$, otherwise we would have $\max([\![\mathsf{C}_2]\!]\eta_1\mathtt{y}) = \max(\eta_1\mathtt{w}) + h_2 = \infty$, 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 $\max([\![\mathsf{C}_1]\!]\eta\mathtt{w}) = \max(\eta\mathtt{w}') + h_1$ and for all $\eta' \in \mathbb{A}$ with $\eta \sqsubseteq \eta'$

$$\max([\![\mathsf{C}_1]\!]\eta'\mathtt{w}) \geq \max(\eta'\mathtt{w}') + h_1 \tag{$\ddagger$}$$

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

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

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

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

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 $\mathtt{x} \in Var$ and $j \in \{0, 1, \ldots, m\}$, if $\max(\eta_{j+1}\mathtt{x}) > (\mathsf{C} + \mathsf{true})^\flat = (\mathsf{C})^\flat$ then there exist $\mathtt{z} \in Var$ and $h \in \mathbb{Z}$ such that $|h| \leqslant (\mathsf{C})^\flat$ 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 $\mathtt{y} \in Var$

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

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

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

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

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

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

$$\sigma \triangleq (\mathtt{y}_0, 0) \to_{h_0} (\mathtt{y}_1, 1) \to_{h_1} \ldots \to_{h_m} (\mathtt{y}_{m+1}, m+1) \tag{3.2}$$

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

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

and we have that:

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

with $|h_j| \leqslant (\mathsf{C})^{\mathtt{b}}$ for $j \in \{i, \ldots, m\}$. Hence we can apply Lemma 3.4 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,

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

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

$$\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{k_2 - k_1} \eta'\mathtt{w}) \geq \max(\eta'\mathtt{w}) + h \tag{3.3}$$

This property (3.3) can be shown by induction on $k_2 - k_1 \geq 1$.

Then, an inductive argument allows us to show that for all $r \in \mathbb{N}$:

$$\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2 - k_1)} \eta_{k_1}\mathtt{w}) \geq \max(\eta_{k_1}\mathtt{w}) + rh \tag{3.4}$$

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

$$
\begin{aligned}
&\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{(r+1)(k_2 - k_1)} \eta_{k_1}\mathtt{w}) = \\
&\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{k_2 - k_1}(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2 - k_1)} \eta_{k_1})\mathtt{w}) \geq \quad [\text{by (3.3) as } \eta_{k_1} \sqsubseteq \llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2 - k_1)} \eta_{k_1}] \\
&\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2 - k_1)} \eta_{k_1}\mathtt{w}) + h \geq \hspace{4.5cm} [\text{by inductive hypothesis}] \\
&\max(\eta_{k_1}\mathtt{w}) + rh + h \geq \max(\eta_{k_1}\mathtt{w}) + (r+1)h
\end{aligned}
$$

However, This would contradict the hypothesis $\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta\mathtt{y} \neq \infty$. In fact the inequality (3.4) would imply

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

Now, from (3.2) 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 $\mu_{j+1} = [\![C + \mathsf{true}]\!]\mu_j$, we have that $\max(\mu_{j+1}\mathbf{y}_{j+1}) \geq \max(\mu_{j+1}\mathbf{y}_j) + h_j$ and thus

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

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

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

contradicting the assumption.

Therefore, the path σ of (3.2) must exist, and consequently

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

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

Let us now show (ii). Given $\eta' \sqsupseteq \eta$ from (3.2) we deduce that for all $j \in \{0, \ldots, m\}$, if we let $\mu_0 = \eta'$ and $\mu_{j+1} = [\![C + \mathsf{true}]\!]\mu_j$, we have that

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

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

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

as desired.   □

Lemma 3.5 provides an effective algorithm for computing the interval semantics of commands. More precisely, given a command C, the corresponding finite set of variables $Var_C \triangleq vars(C)$, and an interval environment $\rho : Var_C \to Int$, we define

$$\max(\rho) \triangleq \max\{\max(\rho(\mathbf{x})) \mid \mathbf{x} \in Var_C\}.$$

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

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

Lemma 3.6. *Let $C \in Imp$ be a command. Then, for all finitely supported $\rho : Var \to Int$, the abstract semantics $\langle C^* \rangle \rho = \bigsqcup_{i \in \mathbb{N}} \langle C \rangle^i(\rho)$ computed in $\mathbb{A}$ and in $\mathbb{A}_{C,\rho}$ coincide.*

Proof. Todo: consequence of Lemma 3.5.   □

Chapter 4

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 η, η' : $\mathit{Var} \to \wp^{\neq\varnothing}(\mathbb{Z})$

$$\bot \dot{\subseteq} \eta$$
$$\eta \dot{\subseteq} \eta' \quad \text{if} \quad \forall \mathbf{x} \in \mathit{Var}.\eta(\mathbf{x}) \subseteq \eta'(\mathbf{x})$$

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

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

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

$$\gamma(\bot) \triangleq \varnothing$$
$$\gamma(\eta) \triangleq \{\rho : \mathit{Var} \to \mathbb{Z} \mid \forall \mathbf{x} \in \mathit{Var}.\rho(\mathbf{x}) \in \eta(\mathbf{x})\}$$

Bibliography

[Bel58] Richard Bellman. "On a routing problem". In: *Quarterly of Applied Mathematics* 16 (1958), pp. 87–90. DOI: 10.1090/qam/102435.

[CC77] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. Los Angeles, California: Association for Computing Machinery, 1977, pp. 238–252. ISBN: 9781450373500. DOI: 10.1145/512950.512973. URL: https://doi.org/10.1145/512950.512973.

[CGR18] Patrick Cousot, Roberto Giacobazzi, and Francesco Ranzato. "Program Analysis Is Harder Than Verification: A Computability Perspective". In: *Computer Aided Verification*. Ed. by Hana Chockler and Georg Weissenbacher. Cham: Springer International Publishing, 2018, pp. 75–95. ISBN: 978-3-319-96142-2.

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

[Gaw+09] Thomas Gawlitza et al. "Polynomial Precise Interval Analysis Revisited". In: *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. Ed. by Susanne Albers, Helmut Alt, and Stefan Näher. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 422–437. ISBN: 978-3-642-03456-5. DOI: 10.1007/978-3-642-03456-5_28. URL: https://doi.org/10.1007/978-3-642-03456-5_28.

[Kön26] Dénes König. "Sur les correspondances multivoques des ensembles". In: *Fundamenta Mathematicae* 8 (1926), pp. 114–134. DOI: 10.4064/fm-8-1-114-134.

[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.

[Min18] Antonie Miné. *Static Inference of Numeric Invariants by Abstract Interpretation*. Université Pierre et Marie Curie, Paris, France, 2018.

[Odi92] Piergiorgio Odifreddi. *Classical recursion theory: The theory of functions and sets of natural numbers*. Elsevier, 1992.

[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.

[Rog87] Hartley Rogers Jr. *Theory of recursive functions and effective computability*. MIT press, 1987.

[Tar55] Alfred Tarski. "A lattice-theoretical fixpoint theorem and its applications." In: (1955).