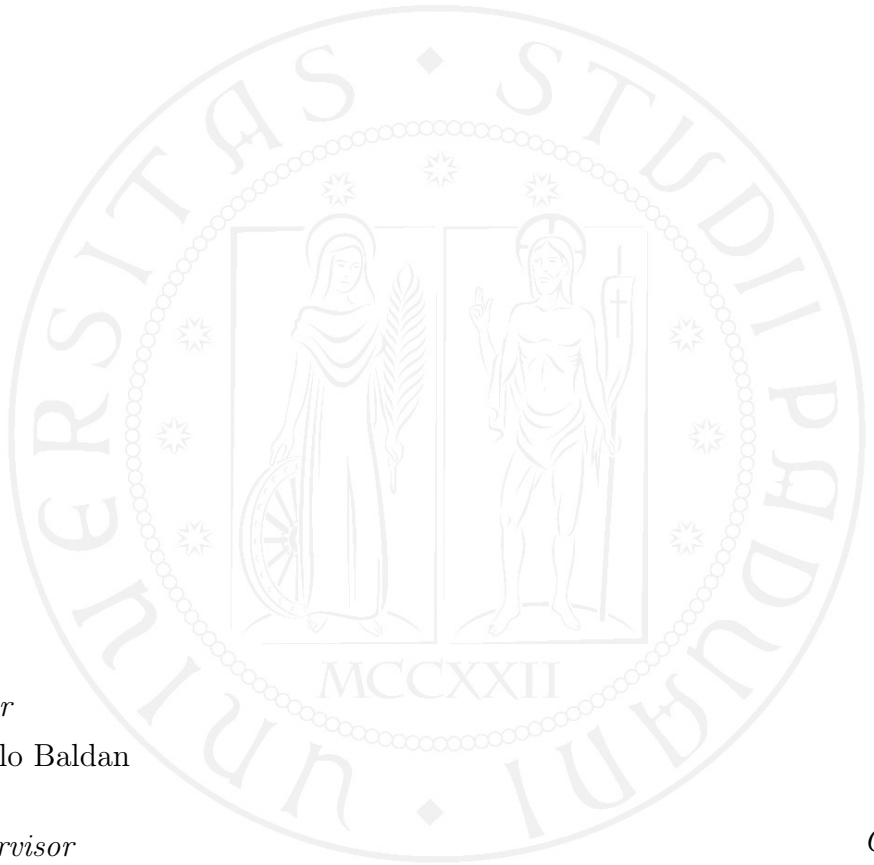
University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master degree in Computer Science

Some decidability questions in abstract program semantics



Supervisor

Prof. Paolo Baldan

Co. Supervisor

Prof. Francesco Ranzato

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

To my family.

Contents

Chapter 1

Background

The following chapter aims to provide context, notation and the important external references for the work that will follow on. It is structured as follows:

- In section 1.1 we make an introduction on the toic, exploring its importance and significance;
- in section 1.2 we will introduce some notation and the important aspect of recursion theory needed to understnd the following chapters;
- finally in section 1.3 we explore order theory and set the notation that we will use in the rest of the thesis to talk about this topic.

1.1 Introduction and related work

Abstract interpretation has faced a difficult challenge since the beginning: the termination of the analyzer. When viewed from another perspective, every interpreter can be considered the most precise possible static analyzer. However, the critical issue is its non-termination. Cousot addresses this problem from the outset in [CC77] by introducing two operators that have since become standard: Widening and narrowing. The former is used to infer properties related to the termination of a loop, albeit at the cost of analysis precision (while still maintaining soundness). However, is it possible to forgo the use of widening operators and still achieve a sound analysis with the same precision as the one utilizing these operators? [Gaw+09] is the first to introduce and demonstrate that this is indeed feasible, presenting an algorithm for calculating fixed-point equation systems using operations and the abstract domain of intervals 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

This first section aims to provide background and terminology for the parts in recursion theory that will follow. More in detail, we'll take some notation from [Cut80] and introduce some new notation based on the same book.

We start with functions: total and partial functions are essential to recursion theory:

Definition 1.1 (Total and partial functions)**.** Let X, Y be two sets. We denote by

$$X \to Y$$

the set of all total functions from X to Y. And by

$$X \hookrightarrow Y$$

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 1.2 (Domain of partial functions)**.** Let $f : X \hookrightarrow Y$. We write $f(x)\downarrow$ to indicate that f is defined on x, and $f(x)\uparrow$ to indicate 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. We therefore define partial recursive and total recursive functions as follows:

Notation 1.1 (partial and total recursive functions)**.** By $\mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ we denote the set of partial recursive functions on natural numbers, while by $\mathbb{N} \overset{r}{\to} \mathbb{N}$ 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 1.3 (Recursively enumerable and recursive sets)**.** A set $A \subseteq \mathbb{N}^k$ is *recursively enumerable* (r.e. or semi-decidable) if $A = dom(f)$ for some $f \in \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$.

A 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 $f \in \mathbb{N} \overset{r}{\to} \mathbb{N}$ s.t.

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

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

Proof. $f : A \overset{r}{\to} B$ 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 an over-approximation of the actual behavior. These mathematical notions, essential for discussions surrounding the Abstract Interpretation formalism, primarily involve order and lattice theory.

Definition 1.4 (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 1.5 (Least upper bound)**.** Let $\langle X, \sqsubseteq \rangle$ be a POSet and let $Z \subseteq X$. We say that $\overline{z} \in Z$ is an *upper bound* of Z if $\forall z \in Z \; z \sqsubseteq \overline{z}$. It is the *least upper bound* of Z (denoted as $\sqcup Z$) if $\forall z'$ upper bounds of Z, $\overline{z} \sqsubseteq z'$.

Definition 1.6 (Greatest lower bound)**.** Let $\langle X, \sqsubseteq \rangle$ be a POSet and let $Z \subseteq X$. We say that $\overline{z} \in Z$ is a *lower bound* of Z if $\forall z \in Z \; \overline{z} \sqsubseteq z$. It is the *greatest lower bound* of Z (denoted as $\sqcap Z$) if $\forall z'$ upper bounds of Z, $z' \sqsubseteq \overline{z}$.

Usually then we are talking about least and greatest lower bound the host 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 1.7 (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 1.8 (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 1.9. $\langle D, \sqsubseteq \rangle$ 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 1.10 (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 1.11 (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*

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

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 $\mathbb{N}$ the set of naturals with the usual order, extended with the top element $+\infty$, s.t. $n \leqslant +\infty$ for all $n \in \mathbb{N}$. We also extend addition and subtraction by letting, for $z \in \mathbb{N}$ it holds that $+\infty + z = +\infty - z = +\infty$ and if $n \leqslant m$ then $n - m = 0$. We focus on the following non-deterministic language.

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

where $\mathtt{x}, \mathtt{y} \in \mathit{Var}$ a finite set of variables of interest, i.e., the variables appearing in the considered program, $S \subseteq \mathbb{N}$ is (possibly empty) *decidable* set of numbers, $a \in \mathbb{N}, b \in \mathbb{N} \cup \{+\infty\}, a \leqslant b, k \in \mathbb{N}$ 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: $\mathsf{Env} \triangleq \{\rho \mid \rho : \mathit{Var} \to \mathbb{N}\}$. For basic expressions $e \in \mathsf{Exp}$ the *concrete semantics* $(\!|\cdot|\!) : \mathsf{Exp} \to \mathsf{Env} \to \mathsf{Env}_\bot$ is inductively defined by:

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

where with $\mathsf{Env}_\bot$ we mean $\mathsf{Env} \cup \{\bot\}$.

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

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

We observe that the semantics we described is additive:

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

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

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 $i \Rightarrow i + 1$:

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

Therefore we can use the inductive hypothesis internally and say

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

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

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

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

Base case:

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

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

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

Inductive cases:

- $\mathsf{C} \equiv \mathsf{C}_1 + \mathsf{C}_2$ therefore we need to show that $\langle \mathsf{com}_1 + \mathsf{C}_2 \rangle X \subseteq \langle \mathsf{com}_1 + \mathsf{C}_2 \rangle Y$

$$\begin{aligned}
\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle X &= \langle \mathsf{C}_1 \rangle X \cup \langle \mathsf{C}_2 \rangle X \\
\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle Y &= \langle \mathsf{C}_1 \rangle Y \cup \langle \mathsf{C}_2 \rangle Y
\end{aligned}$$

by inductive hypothesis both $\langle \mathsf{C}_1 \rangle X \subseteq \langle \mathsf{C}_1 \rangle Y$ and $\langle \mathsf{C}_2 \rangle X \subseteq \langle \mathsf{C}_2 \rangle Y$ and therefore $\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle X \subseteq \langle \mathsf{C}_1 + \mathsf{C}_2 \rangle Y$.

- $\mathsf{C} \equiv \mathsf{C}_1; \mathsf{C}_2$ therefore we need to show that $\langle \mathsf{com}_1; \mathsf{C}_2 \rangle X \subseteq \langle \mathsf{com}_1; \mathsf{C}_2 \rangle Y$

$$\begin{aligned}
\langle \mathsf{C}_1; \mathsf{C}_2 \rangle X &= \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X) \\
\langle \mathsf{C}_1; \mathsf{C}_2 \rangle Y &= \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle Y)
\end{aligned}$$

By induction hypothesis $\langle \mathsf{C}_1 \rangle X \subseteq \langle \mathsf{C}_1 \rangle Y$, and by induction hypothesis again $\langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X) \subseteq \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle Y)$ which means $\langle \mathsf{C}_1; \mathsf{C}_2 \rangle X \subseteq \langle \mathsf{C}_1; \mathsf{C}_2 \rangle Y$.

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

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

    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

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

□

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 $f^{n+1} X = X \cup \langle \mathsf{C} \rangle (f^n X)$.

$$\begin{aligned}
\langle \mathsf{fix}(\mathsf{C}) \rangle X = \mathrm{lfp}(f) &= \bigcup \{ f^n \perp \mid n \in \mathbb{N} \} && \text{by fixpoint theorem (1.1)} \\
&= \bigcup_{i \in \mathbb{N}} \left(X \cup \langle \mathsf{C} \rangle^i X \right) && \text{by definition} \\
&= X \cup (X \cup \langle \mathsf{C} \rangle X) \cup \left(X \cup \langle \mathsf{C} \rangle X \cup \langle \mathsf{C} \rangle^2 X \right) \cup \ldots \\
&= \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i X \\
&= \langle \mathsf{C}^* \rangle X
\end{aligned}$$

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 $\langle \mathsf{C} \rangle \rho$ instead of $\langle \mathsf{C} \rangle \{\rho\}$.

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.

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

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*: $\mathsf{State} \triangleq \mathrm{Imp} \times \mathsf{Env}$ 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 relation $\rightarrow$;
- $\rightarrow^*$ is the reflexive and transitive closure of the relation $\rightarrow$.

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* $\mathsf{C} \in \mathit{Imp}, X \in 2^{\mathsf{Env}}$

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

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

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

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

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

□

Note 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 $\mathsf{C} \in \text{Imp}, \rho \in \mathsf{Env}$. We say C *partially halts* on ρ when there is at least one derivation sequence of finite length in the transition system starting with $\langle \mathsf{C}, \rho \rangle$ and ending 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 $\mathsf{C} \in \text{Imp}, \rho \in \mathsf{Env}$. We say$\mathsf{C}$ *partially loops* on ρ when there is at least one derivation sequence of infinite length in the transition system tarting from $\langle \mathsf{C}, \rho \rangle$:

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

Dually

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

a program *universally halts* on ρ iff there is no infinite derivation sequence starting from $\langle \mathsf{C}, \rho \rangle$ 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 $\mathsf{TS}(\langle \mathsf{C}, \rho \rangle)$ 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

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

it universally halts, since $\forall \rho \in \mathsf{Env}, \rho \neq \bot$

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

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

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

The program never halts on $\forall \rho \in \mathsf{Env}$ s.t. $\rho(\mathtt{x}) \geqslant 0$. In fact in these cases it builds the transition system in figure 2.1, where the infinite derivation sequence

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

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

is always present.

$$\langle (\mathtt{x} \geqslant 0; \mathtt{x{+}{+}})^*; \mathtt{x} < 0, \rho\rangle \longrightarrow \langle \mathtt{x} < 0, \rho\rangle \not\to$$
$$\downarrow$$
$$\langle \mathtt{x} \geqslant; 0\mathtt{x{+}{+}}; (\mathtt{x} \geqslant 0; \mathtt{x{+}{+}})^*; \mathtt{x} < 0, \rho\rangle$$
$$\downarrow^*$$
$$\langle (\mathtt{x} \geqslant 0; \mathtt{x{+}{+}})^*; \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + 1]\rangle \longrightarrow \langle \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + 1]\rangle \not\to$$
$$\downarrow^*$$
$$\langle (\mathtt{x} \geqslant 0; \mathtt{x{+}{+}})^*; \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + k]\rangle \longrightarrow \langle \mathtt{x} < 0, \rho[\mathtt{x} \mapsto \rho(\mathtt{x}) + k]\rangle \not\to$$

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

Example 2.3. Consider the program

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

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

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

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

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

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

Lemma 2.3 (Decomposition lemma)**.** *If $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^k \rho''$, then there exists a state ρ' and a natural number k_1, k_2 s.t. $\langle \mathsf{C}_1, \rho\rangle \to^{k_1} \rho'$ and $\langle \mathsf{C}_2, \rho'\rangle \to^{k_2} \rho''$, where $k_1 + k_2 = k$*

Corollary 2.1. *If $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^* \rho''$ then $\exists \rho'$ s.t. $\langle \mathsf{C}_1, \rho\rangle \to^* \rho'$ and $\langle \mathsf{C}_2, \rho'\rangle \to^* \rho''$.*

Lemma 2.4 (Composition lemma)**.** *If $\langle \mathsf{C}_1, \rho\rangle \to^k \rho'$ then $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^k \langle \mathsf{C}_2, \rho'\rangle$*

Corollary 2.2. *If $\langle \mathsf{C}_1, \rho\rangle \to^* \rho'$ then $\langle \mathsf{C}_1; \mathsf{C}_2, \rho\rangle \to^* \langle \mathsf{C}_2, \rho'\rangle$.*

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

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

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

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

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

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

2.4 Functions in Imp

Last section defined the language we are working with (the Imp language), its semantics and its transition system. Building upon those elements, we now present the first properties of the language. More in detail, 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 properties from well known computability results, without proving them from scratch. We can do this by encoding partial recursive functions into Imp programs. We therefore start by better describing what we mean by partially recursive functions:

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:

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

(b) the successor function

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

(c) the projection function

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

and is closed under

(1) composition: given a function $f : \mathbb{N}^k \overset{r}{\hookrightarrow} \mathbb{N}$ and functions $g_1, \ldots, g_k : \mathbb{N}^n \overset{r}{\hookrightarrow} \mathbb{N}$ 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 $\mathtt{C}$. 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. We say that f is Imp computable if

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

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) from Definition 2.8.

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. □

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. □

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*

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

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. □

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

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

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

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

The final step is to recall the Rice theorem from [Ric53] and the definition of saturated sets:

Definition 2.11 (Saturated set)**.** $A \subseteq \mathbb{N}$ is *saturated* (or *extensional*) is for all $x, y \in \mathbb{N}$

$$x \in A \wedge \varphi_x = \varphi_y \Rightarrow y \in A$$

In other words a set is saturated if it contains all the numbers that encode for programs that compute functions with some properties. Rice's theorem links extensional sets and decidability:

Theorem 2.1 (Rice's theorem)**.** *Let $A \subseteq \mathbb{N}$, $A \neq \varnothing, A \neq \mathbb{N}$ be a saturated set. Then A is not recursive.*

Meaning that deciding weather a program is in some saturaed set, i.e., the program has some extensional property, is impossible. 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., $\langle \mathsf{C} \rangle \rho = \varnothing$) is undecidable.*

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

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

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

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 if we were able to establish whether $\langle \mathsf{C} \rangle X$ is finite for some program $\mathsf{C} \in \mathrm{Imp}$ and some initial states $X \in \mathbb{C}$, we could decide whether $\langle \mathsf{C}, \rho \rangle \Downarrow$ for all $\rho \in X$, which instead is known to be undecidable. 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 $\mathsf{D} \in Imp_s$, 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). $|\mathsf{TS}(\langle \mathsf{D}, \rho \rangle)| < \infty$ *for all* $\rho \in X$*.*

where by $\mathsf{TS}(\langle \mathsf{D}, \rho \rangle)$ we mean the set of all derivation sequences starting from $\langle \mathsf{D}, \rho \rangle$ in the transition system.

Proof. By induction on the program D:

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

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

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

(iii). Notice that $\forall \rho \in X$ either by the expr rule $\langle \mathsf{e}, \rho \rangle \to (\!| \mathsf{e} |\!) \rho$ or $\langle \mathsf{e}, \rho \rangle \not\to$ therefore

$$|\mathsf{TS}(\langle \mathsf{e}, \rho \rangle)| \leqslant |X| < \infty$$

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 $\forall \rho \in X \ \langle \mathsf{D}_1, \rho \rangle \downdownarrows$ and $\langle \mathsf{D}_2, \rho \rangle \downdownarrows$. By sum_1 rule $\langle \mathsf{D}_1 + \mathsf{D}_2, \rho \rangle \to \langle \mathsf{D}_1, \rho \rangle$ and by sum_2 $\langle \mathsf{D}_1 + \mathsf{D}_2, \rho \rangle \to \langle \mathsf{D}_2, \rho \rangle$. Therefore $\langle \mathsf{D}_1 + \mathsf{D}_2 \rangle \rho \downdownarrows$.
   (iii). For the latter argument, since both $\langle \mathsf{D}_1 \rangle \rho$ and $\langle \mathsf{D}_2 \rangle \rho$ are finite and composed of finite derivation sequences $|\mathsf{TS}(\langle \mathsf{D}_1 + \mathsf{D}_2, \rho \rangle)| < \infty$.
2. $\mathsf{D} \equiv \mathsf{D}_1; \mathsf{D}_2$, therefore
   (i). $\langle \mathsf{D}_1; \mathsf{D}_2 \rangle X = \langle \mathsf{D}_2 \rangle (\langle \mathsf{D}_1 \rangle X)$. By inductive hypothesis $\langle \mathsf{D}_1 \rangle X = Y$. By inductive hypothesis again $\langle \mathsf{D}_2 \rangle Y$ is finite;
   (ii). by inductive hypothesis both $\forall \rho \in X \ \langle \mathsf{D}_1, \rho \rangle \downdownarrows$ and $\forall \rho' \in Y \ \langle \mathsf{D}_2, \rho' \rangle \downdownarrows$, therefore by composition lemma $\langle \mathsf{D}_1; \mathsf{D}_2, \rho \rangle \downdownarrows$
   (iii). by inductive hypothesis both $|\mathsf{TS}(\langle \mathsf{D}_1, \rho \rangle)| < \infty$ and $|\mathsf{TS}(\langle \mathsf{D}_2, \rho' \rangle)| < \infty \ \forall \rho \in X, \rho' \in \langle \mathsf{D}_1 \rangle X$. For all derivation sequences starting from $\langle \mathsf{D}_1, \rho \rangle$ where

$$\langle \mathsf{D}_1, \rho \rangle \to^* \rho'$$

   with $\rho' \in \langle \mathsf{D}_1 \rangle X$ we can apply the composition lemma and state that

$$\langle \mathsf{D}_1; \mathsf{D}_2, \rho \rangle \to^* \langle \mathsf{D}_2, \rho' \rangle \quad \forall \rho \in X$$

   from there we can notice that since $|\langle \mathsf{D}_2, \rho' \rangle| < \infty$ then $|\langle \mathsf{D}_1; \mathsf{D}_2, \rho' \rangle| < \infty$

□

In order to prove that finiteness is undecidable we need the following Lemma:

Lemma 2.10. *Let* $\mathsf{D} \in Imp_s$ *and* $\rho \in \mathsf{Env}$. *If*

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

then

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

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

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

- if $j = 0$ then we want to show that $\langle \mathsf{D} \rangle^{k+1} \rho \subseteq \cup_{i=0}^{k} \langle \mathsf{D} \rangle^i \rho$, 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

$$\begin{aligned}
\bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^i \rho &= \bigcup_{i=0}^{k+1} \langle \mathsf{D} \rangle^i \rho && \text{since by (2.2) } \langle \mathsf{D} \rangle^{k+1} \rho \subseteq \cup_{i=0}^{k} \langle \mathsf{D} \rangle^i \rho \\
&= \rho \cup \bigcup_{i=1}^{k+1} \langle \mathsf{D} \rangle^i \rho \\
&= \rho \cup \langle \mathsf{D} \rangle \left(\bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^i \rho \right) && \text{by additivity of } \langle \mathsf{D} \rangle
\end{aligned}$$

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

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

□

We also need to recall Konig's Lemma from [Kön26]:

Lemma 2.11 (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.*

With Lemma 2.10 and Lemma 2.11 we can prove the following.

Lemma 2.12. *Given $\mathsf{D} \in Imp_s$, and $\rho \in \mathsf{Env}$, the predicate "$\langle \mathsf{D}^* \rangle \rho$ is finite" is undecidable.*

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

- Suppose that $\langle \mathsf{D}^* \rangle \rho$ is infinite, then we can observe that because Lemma 2.10

$$\forall k \in \mathbb{N} \quad \langle \mathsf{D} \rangle^{k+1} \rho \not\subseteq \bigcup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho \tag{2.4}$$

Otherwise $\langle \mathsf{D}^* \rangle \rho \subseteq \bigcup_{i \in \mathbb{N}} \langle \mathsf{D} \rangle^{i} \rho$ and we would contraddict the hypothesis of $\langle \mathsf{D}^* \rangle \rho$ being infinite. 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 $\rho' \rightarrow^D \rho''$ if $\langle \mathsf{D}, \rho' \rangle \rightarrow^* \rho''$. 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) since $\forall k \in \mathbb{N}\ \langle \mathsf{D} \rangle^{k+1} \rho \not\subseteq \cup_{i=0}^{k} \langle \mathsf{D} \rangle^{i} \rho$ by (2.4), each level Y_k is non empty. Each level is also finite because of Lemma 2.1.(i). Therefore there is an infinite quantity of levels, where each node has a finite quantity of children.

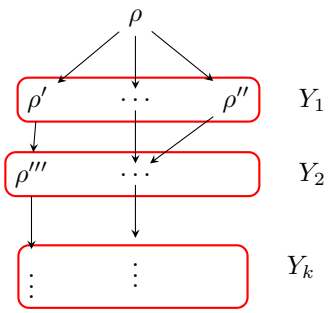


Figure 2.2: Example of $\rightarrow^D$ relations between elements of Env.

what is left to do is show that there is a derivation sequence from $\langle \mathsf{D}^*, \rho \rangle$ of infinite length. We can do this by using König's Lemma 2.11 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. Observe that for all $\rho', \rho'' \in \mathsf{Env}$ s.t. $\rho' \rightarrow^D \rho''$ since $\langle \mathsf{D}, \rho' \rangle \rightarrow^* \rho''$ we can apply Corollary 2.2 of Lemma 2.4 and derive 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

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

Therefore, there exists an infinite derivation sequence

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

which means $\langle \mathsf{D}^*, \rho \rangle \uparrow$ and therefore $\langle \mathsf{D}^*, \rho \rangle \downarrow\downarrow$ is false.

- if instead $\langle \mathsf{D}^* \rangle \rho$ 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 $\langle \mathsf{D}^*, \rho \rangle \downarrow\downarrow \iff$ 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 $\langle \mathsf{D}, \rho \rangle \downarrow\downarrow$ 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

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

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

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

  is part of the transition system for $\langle \mathsf{D}, \rho \rangle$, and therefore $\langle \mathsf{D}^*, \rho \rangle \downarrow\downarrow$ 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 $\langle \mathsf{D}, \rho \rangle \downarrow\downarrow$. 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 would contraddict the hypothesis. Therefore the derivation sequence has the shape

  $$\langle \mathsf{D}^*, \rho \rangle \rightarrow^* \langle \mathsf{D}^*, \rho_1 \rangle \rightarrow^* \langle \mathsf{D}^*, \rho_2 \rangle \rightarrow^* \langle \mathsf{D}^*, \rho_3 \rangle \rightarrow^* \dots$$

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

To conclude we can observe that there is a finite amount of reachable states from $\langle \mathsf{D}^*, \rho \rangle$. Where by reachable we mean that there exists some derivation sequence ending up in that state.

We can notice that starting from any state $\langle \mathsf{D}^*, \rho' \rangle$ with $\rho' \in \langle \mathsf{D}^* \rangle \rho$ we have 2 possibilities:

- we either apply the star_{fix}rule, resulting in a finite derivation sequence

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

  and therefore in a finite number of reached states;
- or we apply the star rule

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

  by lemma 2.9 we know that $\langle \mathsf{D}, \rho' \rangle \downdownarrows$ and $|\mathsf{TS}(\langle \mathsf{D}, \rho' \rangle)| < \infty$, therefore there is a finite number of environments ρ'' s.t. $\langle \mathsf{D}, \rho' \rangle \to^* \rho''$. For each one of them we can use the composition lemma and observe that

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

  Ending up in a state $\langle \mathsf{D}^*, \rho'' \rangle$ where we can apply the same reasoning

Therefore starting from any state $\langle \mathsf{D}^*, \rho' \rangle$ with $\rho' \in \langle \mathsf{D}^* \rangle \rho$ (in particular ρ), we either terminate our derivation sequence or we end up in some state $\langle \mathsf{D}^*, \rho' \rangle$ again, with $\rho' \in \langle \mathsf{D}^* \rangle \rho$. Since there is a finite amount of states $\rho' \in \langle \mathsf{D}^* \rangle \rho$, the number of reachable states from $\langle \mathsf{D}^*, \rho \rangle$ is finite.

□

Chapter 3

Abstract domains

In the following chapters we present two domains that we will discuss: the *interval* domain and the *non-relational* collecting domain. The two domains are in the class of *non-relational* domains, meaning that they do not rapresent the relation between variables. We are interested in these two domains as the properties that we will discuss in Chapter 4 will apply to these domains, with some restrictions that we will discuss later. This chapter's sections are organized as follows:

Expand

§3.1 Will talk about the interval domain, with its characterization in §3.1.1 and the domain properties in §3.1.2.

§3.2 Will talk about the non-relational collecting abstraction, firstly introducing the definition of the non-relational collecting domain in §3.2.1 and finally by showing some properties of the abstraction in §3.2.2

3.1 Intervals domain

Interval analysis are among the most well known abstract interpretation standard abstract domains. They are generally studied as simple non-relational domains, as intervals are not able to capture the relation between variables occurring in the program. The following chapter aims to prove the fact that interval analysis is decidable without a widening operator, i.e., infinite ascending chains can be decided.

3.1.1 Definition

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

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

set of integer intervals. We write $\top$ instead of $[-\infty, +\infty]$

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

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

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 $Int_* \triangleq Int \setminus \{\bot^\sharp\}$. The abstract domain $\mathbb{I}$ for program analysis is the variable-wise lifting of Int:

$$\mathbb{I} \triangleq (Var \to Int_*) \cup \{\bot^\sharp\}$$

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

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

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

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

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

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

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

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

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

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

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

Definition 3.6 (Abstract operations)**.** We define sound abstract lub and glb operations in the $\mathbb{I}$ domain:

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

And sound abstract arithmetical operations:

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

Definition 3.7 (Interval sharpening)**.** For a nonempty interval $[a,b] \in \mathit{Int}$ and $c \in \mathbb{N}$, we define two operations raising $\uparrow$ the lower bound to c and lowering $\downarrow$ the upper bound to c, respectively:

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

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

Definition 3.8 (Interval addition and subtraction)**.** For a nonempty interval $[a,b] \in \mathit{Int}$ and $c \in \mathbb{N}$ define $[a,b] \pm c \triangleq [a \pm c, b \pm c]$ (recall that $\pm\infty + c = \pm\infty - c = \pm\infty$). □

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

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

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

Definition 3.9 (Interval semantics)**.** The *interval semantics* of Imp is the strict (i.e., preserving $\bot$) extension of the following function $[\![\cdot]\!] : \text{Imp} \to \mathbb{I} \to \mathbb{I}$. For all $\eta : \mathit{Var} \to \mathit{Int}_*$,

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

Notice that the filtering in Definition 3.9 is not the best correct approximation. In particular the filtering instruction $\mathtt{x} \in S$ is performed first by abstracting the numerical set $S \subseteq \mathbb{N}$ with α_{Int} and then by computing the greatest lower bound with $\eta(\mathtt{x})$. Best correct approximation would consist in the opposite approach: compute the concrete $\gamma(\eta\mathtt{x})$ and subsequently compute the meet in the $2^{\mathbb{N}}$ domain:

$$[\![\mathtt{x} \in S]\!]\eta = \begin{cases} \eta[\mathtt{x} \mapsto \alpha_{\mathit{Int}}(\gamma(\eta\mathtt{x}) \cap S)] & \text{if } \gamma(\eta\mathtt{x}) \cap S \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

This however would introduce a problem in proving Lemma 4.3, as Axiom **??** would not be respected. Notice that this approximation coincides in fact with the best correct approximation in two important cases: $\mathtt{x} \leqslant k$ and $\mathtt{x} \geqslant k$. These two cases are widely used in programming, namely we can cite the fact that they are part of the nasa programming guidelines on writing analyzer-friendly code as stated in [Hol06].

3.1.2 Properties

We can immediately see how in the abstract interval domain, the semantics of the Kleene star and the fixpoint operator is not the same. This intuitively happens because the kleene star is the least opper bound of a chain of interals, while the fix operator keeps iterating over least upper bounds.

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:

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

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

Lemma 3.1 ($\mathsf{fix}(\mathsf{C})$ **is syntactic sugar**). *For all* η, $[\![\mathsf{fix}(\mathsf{C})]\!]\eta = [\![(\mathsf{true} + \mathsf{C})^*]\!]\eta$.

Proof. Let us first show by induction that

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

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

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

Let us also show that:

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

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

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

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

Then, we have that:

$$\begin{aligned}
[\![\mathsf{fix}(\mathsf{C})]\!]\eta &= \\
\mathrm{lfp}\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu) &= && \text{By } (\diamond) \\
\mathrm{lfp}\lambda\mu.(\eta \sqcup \mu \sqcup [\![\mathsf{C}]\!]\mu) &= && \text{By Knaster-Tarski Theorem} \\
\bigsqcup_{i\in\mathbb{N}} (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^i\bot &= \\
\bot \sqcup \bigsqcup_{i\in\mathbb{N}} (\eta \sqcup \mathsf{true} \sqcup [\![\mathsf{C}]\!])^{i+1}\bot &= && \text{By (4.2)} \\
\bigsqcup_{i\in\mathbb{N}} (\mathsf{true} \sqcup [\![\mathsf{C}]\!])^i\eta &= \\
[\![(\mathsf{true} + \mathsf{C})^*]\!]\eta. &
\end{aligned}$$

□

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, $[\![\mathsf{C}]\!]\eta$ 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 Non relational collecting

In previous we saw how by using bound we can produce a sound abstraction in the interval domain that coincides with the abstraction traditionally made with the use of widening and narrowing operators, in particular with lemma 4.3. In this chapter we argue that the same procedure can be used in with a general non-relational collecting abstraction. We first define what we mean by non relational collecting abstraction and later see how we can adapt lemma 4.3 to this purpouse and produce a sonud abstraction without widening and narrowing operators.

3.2.1 Definition

We first define *non-relational collecting* analysis the the Imp language in a standard way.

Definition 3.10 (Non Relational Collecting domain)**.** Let $NRel \triangleq 2^{\mathbb{N}} \setminus \{\varnothing\}$ and

$$\mathsf{Env}^{\mathsf{c}} \triangleq \{\eta \mid \eta : Var \to NRel\} \cup \{\bot\}$$

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

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

In order to have a proper galois connection we also define abstraction and concretization maps:

Definition 3.11 (Non relational collecting abstraction)**.** we define the *non relational abstraction* map

$$\alpha : \langle 2^{\mathsf{Env}}, \subseteq \rangle \to \langle \mathsf{Env}^{\mathsf{c}}, \dot{\subseteq} \rangle$$

as follows:

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

Definition 3.12 (Non relational concretization)**.** We define the *concretization map*

$$\gamma : \langle \mathsf{Env}^{\mathsf{c}}, \dot{\subseteq} \rangle \to \langle 2^{\mathsf{Env}}, \subseteq \rangle$$

as follows:

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

With an abstraction and concretization map we have a galois connection, where we also define the least upper bound and the greatest lower bound on elements of $\mathbb{C}^{\mathsf{c}}$: finish

Definition 3.13 (lub and glb in $\mathbb{C}^{\mathsf{c}}$)**.** Let $\eta, \vartheta \in \mathbb{C}^{\mathsf{c}}$. we define the $\sqcup$ and $\sqcap$ operations:

$$\begin{aligned} \eta \sqcup \vartheta &= \sigma && \text{if } \sigma\mathtt{x} = \eta\mathtt{x} \cup \vartheta\mathtt{x} \quad \forall \mathtt{x} \in Var \\ \eta \sqcap \vartheta &= \sigma && \text{if } \sigma\mathtt{x} = \eta\mathtt{x} \cap \vartheta\mathtt{x} \quad \forall \mathtt{x} \in Var \end{aligned}$$

finally we can define non relational collecting analysis.

Definition 3.14 (Non-Relational collecting semantics)**.** The *Non-Relational collecting semantics* of Imp, is the strict (i.e., preserving $\bot$) extension of the following function $\langle\!\langle \cdot \rangle\!\rangle : \text{Imp} \to \mathbb{C}^{\mathsf{c}} \to \mathbb{C}^{\mathsf{c}}$. For all $\eta \in \mathsf{Env}^{\mathsf{c}}$,

$$\begin{aligned} \langle\!\langle \mathtt{x} \in S \rangle\!\rangle \eta &\triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \cap S] & \text{if } \eta(\mathtt{x}) \cap S \neq \bot \\ \bot & \text{otherwise} \end{cases} \\ \langle\!\langle \mathsf{true} \rangle\!\rangle \eta &\triangleq \eta \\ \langle\!\langle \mathsf{false} \rangle\!\rangle \eta &\triangleq \bot \\ \langle\!\langle \mathtt{x} := k \rangle\!\rangle \eta &\triangleq \eta[\mathtt{x} \mapsto \{k\}] \\ \langle\!\langle \mathtt{x} := \mathtt{y} + k \rangle\!\rangle \eta &\triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) + k] \\ \langle\!\langle \mathtt{x} := \mathtt{y} - k \rangle\!\rangle \eta &\triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) - k] \\ \langle\!\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle\!\rangle \eta &\triangleq \langle\!\langle \mathsf{C}_1 \rangle\!\rangle \eta \sqcup \langle\!\langle \mathsf{C}_2 \rangle\!\rangle \eta \\ \langle\!\langle \mathsf{C}_1; \mathsf{C}_2 \rangle\!\rangle \eta &\triangleq \langle\!\langle \mathsf{C}_2 \rangle\!\rangle (\langle\!\langle \mathsf{C}_1 \rangle\!\rangle \eta) \\ \langle\!\langle \mathsf{C}^* \rangle\!\rangle \eta &\triangleq \bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta) \\ \langle\!\langle \mathsf{fix}(\mathsf{C}) \rangle\!\rangle \eta &\triangleq \text{lfp}(\lambda \mu. (\eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle \mu)) \end{aligned}$$

3.2.2 Properties

We can notice that the semantics we just defined has some properties similar to the interval semantics but also the concrete collecting semantics from Definition 2.2. The main property is additivity, which we lost with the interval semantics and got back with the non-relational collecting:

Lemma 3.2 (Additivity)**.** *Let* $\eta, \vartheta \in \mathbb{C}^{\mathsf{c}}$, $\mathsf{C} \in \mathit{Imp}$ *then*

$$\langle\!\langle \mathsf{C} \rangle\!\rangle (\eta \sqcup \vartheta) = (\langle\!\langle \mathsf{C} \rangle\!\rangle \eta) \sqcup (\langle\!\langle \mathsf{C} \rangle\!\rangle \vartheta)$$

Proof. We can work by induction on C:

Base cases:

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

$$\begin{aligned}\langle\!\langle \mathtt{x} \in S \rangle\!\rangle(\eta \sqcup \vartheta) &= (\eta \sqcup \vartheta)[\mathtt{x} \mapsto (\eta \sqcup \vartheta)\mathtt{x} \cap S] \\ &= (\eta \sqcup \vartheta)[\mathtt{x} \mapsto (\eta\mathtt{x} \cap S) \cup (\vartheta\mathtt{x} \cap S)] \\ &= (\eta[\mathtt{x} \mapsto (\eta\mathtt{x} \cap S)]) \sqcup (\vartheta[\mathtt{x} \mapsto \vartheta\mathtt{x} \cap S]) \\ &= \langle\!\langle \mathtt{x} \in S \rangle\!\rangle \eta \sqcup \langle\!\langle \mathtt{x} \in S \rangle\!\rangle \vartheta\end{aligned}$$

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

$$\begin{aligned}\langle\!\langle \mathtt{x} := k \rangle\!\rangle(\eta \sqcup \vartheta) &= (\eta \sqcup \vartheta)[\mathtt{x} \mapsto \{k\}] \\ &= (\eta[\mathtt{x} \mapsto \{k\}]) \sqcup (\vartheta[\mathtt{x} \mapsto \{k\}]) \\ &= \langle\!\langle \mathtt{x} := k \rangle\!\rangle \eta \sqcup \langle\!\langle \mathtt{x} := k \rangle\!\rangle \vartheta\end{aligned}$$

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

$$\begin{aligned}\langle\!\langle \mathtt{x} := \mathtt{y} + k \rangle\!\rangle(\eta \sqcup \vartheta) &= (\eta \sqcup \vartheta)[\mathtt{x} \mapsto \mathtt{y} + k] \\ &= (\eta[\mathtt{x} \mapsto \mathtt{y} + k]) \sqcup (\vartheta \mapsto \mathtt{y} + k) \\ &\langle\!\langle \mathtt{x} := \mathtt{y} + k \rangle\!\rangle \eta \sqcup \langle\!\langle \mathtt{x} := \mathtt{y} + k \rangle\!\rangle \vartheta\end{aligned}$$

- $\mathsf{C} \equiv \mathtt{x} := \mathtt{y} - k$. Is analogous to the latter case.

Recursive cases:

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

$$\begin{aligned}\langle\!\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle\!\rangle(\eta \sqcup \sigma) &= \langle\!\langle \mathsf{C}_1 \rangle\!\rangle(\eta \sqcup \sigma) \sqcup \langle\!\langle \mathsf{C}_2 \rangle\!\rangle(\eta \sqcup \sigma) && \text{by definition} \\ &= \langle\!\langle \mathsf{C}_1 \rangle\!\rangle \eta \sqcup \langle\!\langle \cup_1 \rangle\!\rangle \vartheta \sqcup \langle\!\langle \mathsf{C}_2 \rangle\!\rangle \eta \sqcup \langle\!\langle \cup_2 \rangle\!\rangle \vartheta && \text{by inductive hypothesis} \\ &= \langle\!\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle\!\rangle \eta \sqcup \langle\!\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle\!\rangle \vartheta\end{aligned}$$

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

$$\begin{aligned}\langle\!\langle \mathsf{C}_1; \mathsf{C}_2 \rangle\!\rangle(\eta \sqcup \sigma) &= \langle\!\langle \mathsf{C}_2 \rangle\!\rangle(\langle\!\langle \mathsf{C}_1 \rangle\!\rangle(\eta \sqcup \vartheta)) \\ &= \langle\!\langle \mathsf{C}_2 \rangle\!\rangle (\langle\!\langle \mathsf{C}_1 \rangle\!\rangle \eta \sqcup \langle\!\langle \cup_1 \rangle\!\rangle \vartheta) && \text{by inductive hypothesis} \\ &= \langle\!\langle \mathsf{C}_2 \rangle\!\rangle(\langle\!\langle \mathsf{C}_1 \rangle\!\rangle \eta) \sqcup \langle\!\langle \mathsf{C}_2 \rangle\!\rangle(\langle\!\langle \mathsf{C}_1 \rangle\!\rangle \vartheta) && \text{by inductive hypothesis}\end{aligned}$$

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

$$\langle\!\langle \mathsf{C}^* \rangle\!\rangle(\eta \sqcup \vartheta) = \bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta \sqcup \vartheta)$$

  What we have to show now is that $\forall i \in \mathbb{N}\ \langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta \sqcup \vartheta) = \langle\!\langle \mathsf{C} \rangle\!\rangle^i \eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle^i \vartheta$. We can show this by induction on i:

– $i = 0$. Then
$$\langle\!\langle \mathsf{C} \rangle\!\rangle^0(\eta \sqcup \vartheta) = \eta \sqcup \vartheta = \langle\!\langle \mathsf{C} \rangle\!\rangle^0 \eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle^0 \vartheta$$
and the statement holds.

– $i \Rightarrow i + 1$. Notice that
$$\begin{aligned}
\langle\!\langle \mathsf{C} \rangle\!\rangle^{i+1}(\eta \sqcup \vartheta) &= \langle\!\langle \mathsf{C} \rangle\!\rangle \left(\langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta \sqcup \vartheta) \right) \\
&= \langle\!\langle \mathsf{C} \rangle\!\rangle (\langle\!\langle \mathsf{C} \rangle\!\rangle^i \eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle^i \vartheta) && \text{by inductive hypothesis} \\
&= \langle\!\langle \mathsf{C} \rangle\!\rangle^{i+1} \eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle^{i+1} \vartheta && \text{by additivity}
\end{aligned}$$

Therefore
$$\begin{aligned}
\langle\!\langle \mathsf{C}^* \rangle\!\rangle(\eta \sqcup \vartheta) &= \textstyle\bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta \sqcup \vartheta) \\
&= \textstyle\bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i (\eta \sqcup \vartheta) \\
&= \textstyle\bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i \eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle^i \vartheta \\
&= \left(\textstyle\bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i \eta \right) \sqcup \left(\textstyle\bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i \vartheta \right) \\
&= \langle\!\langle \mathsf{C}^* \rangle\!\rangle \eta \sqcup \langle\!\langle \mathsf{C}^* \rangle\!\rangle \vartheta
\end{aligned}$$

□

Again, because of additivity we can notice that the analysis of the fixpoint and the kleene star is the same. Let $f = \lambda\mu.(\eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle \mu)$

$$\begin{aligned}
\langle\!\langle \mathsf{fix}(\mathsf{C}) \rangle\!\rangle \eta &= \mathrm{lfp}(f) \\
&= \bigsqcup_{i \in \mathbb{N}} \{ f^n \bot \mid n \in \mathbb{N} \} && \text{by fixpoint theorem 1.1} \\
&= \eta \sqcup (\eta \sqcup \langle\!\langle \mathsf{C} \rangle\!\rangle \eta) \sqcup \ldots && \text{by definition} \\
&= \bigsqcup_{i \in \mathbb{N}} \langle\!\langle \mathsf{C} \rangle\!\rangle^i \eta \\
&= \langle\!\langle \mathsf{C}^* \rangle\!\rangle \eta
\end{aligned}$$

therefore we will omit one of the two cases based on preference and ease of reading, but in reality they are the same

Chapter 4

Program bounds and analysis termination

In this chapter we argue that for the language Imp the abstract semantics is computable in finite time without widening for abstract domains with some properties.

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 if we consider the intervals abstract domain, 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. First let's formalize the problem we want to solve.

Problem 4.1 (Analysis termination)**.** Given $\mathsf{C} \in \text{Imp}$, $\eta \in \mathbb{A}$, decide: $[\![\mathsf{C}]\!]\eta =^{?} \top$

To do so we present a novel technique, based on the idea of *bounds*. Each program is associated to a bound, an ideal value above which for each variable we cannot guarantee convergence, and therefore we can safely assume that the program diverges. First, given a program, we associate each variable with a *single bound*, which captures both both an *upper bound* and a *lower bound* for the variable. The rough idea is that, whenever a variable is within its bound, the behavior of the program with respect to that variable becomes stable.

4.1 Analysis assumptions

Given some abstract domain $\mathbb{A}$ over some domain $\mathbb{D}$ (e.g. $\mathbb{A}$ over *Int*) we define an *analysis over* $\mathbb{A}$ as the function $[\![\cdot]\!] : \text{Imp} \to \mathbb{A} \to \mathbb{A}$, which maps each Imp command to a continuous function over the $\mathbb{A}$ lattice. The analysis is inductively defined as follows:

Definition 4.1 (Abstract interpretation)**.** The *interval semantics* of Imp is the strict (i.e., preserving $\bot$) extension of the following function $[\![\cdot]\!] : \text{Imp} \to \mathbb{A} \to \mathbb{A}$. For all $\eta : \mathit{Var} \to \mathbb{D}_*$,

$$[\![\mathtt{x} \in S]\!]\eta \triangleq \begin{cases} \eta[\mathtt{x} \mapsto \eta(\mathtt{x}) \sqcap \alpha_{\mathit{Int}}(S)] & \text{if } \eta(\mathtt{x}) \sqcap \alpha(S) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\mathtt{x} := k]\!]\eta \triangleq \eta[\mathtt{x} \mapsto \alpha(k)]$$

$$[\![\mathtt{x} := \mathtt{y} + k]\!]\eta \triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) + k]$$

$$[\![\mathtt{x} := \mathtt{y} - k]\!]\eta \triangleq \eta[\mathtt{x} \mapsto \eta(\mathtt{y}) - k]$$

$$[\![\mathsf{C}_1 + \mathsf{C}_2]\!]\eta \triangleq [\![\mathsf{C}_1]\!]\eta \sqcup [\![\mathsf{C}_2]\!]\eta$$

$$[\![\mathsf{C}_1; \mathsf{C}_2]\!]\eta \triangleq [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]\eta)$$

$$[\![\mathsf{C}^*]\!]\eta \triangleq \bigsqcup_{i \in \mathbb{N}} [\![\mathsf{C}]\!]^i(\eta)$$

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

Where by $\mathbb{D}_*$ we mean $\mathbb{D} \setminus \{\varnothing\}$.

Notice that the filtering in Definition 3.9 is not the best correct approximation. In particular the filtering instruction $\mathtt{x} \in S$ is performed first by abstracting the numerical set $S \subseteq \mathbb{N}$ with α and then by computing the greatest lower bound with $\eta(\mathtt{x})$. Best correct approximation would consist in the opposite approach: compute the concrete $\gamma(\eta\mathtt{x})$ and subsequently compute the meet in the domain $2^{\mathbb{N}}$:

$$[\![\mathtt{x} \in S]\!]\eta = \begin{cases} \eta[\mathtt{x} \mapsto \alpha_{\mathit{Int}}(\gamma(\eta\mathtt{x}) \cap S)] & \text{if } \gamma(\eta\mathtt{x}) \cap S \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

This however would introduce a problem in proving Lemma 4.3, as Axiom **??** would not be respected if we compute the analysis using as abstract domain $\mathbb{C}^{\mathsf{c}}$. Notice that this approximation coincides in fact with the best correct approximation in one important case: $\mathtt{x} \leqslant k$. This case is widely used in programming and is already considered a good way of writing analizer-friendly code. to support this thesis we can cite the fact that it is part of the nasa programming guidelines on writing analyzer-friendly code, as reported in [Hol06].

4.2 Program bounds

Definition 4.2 (**Program bound**)**.** The *bound* associated with a command $\mathsf{C} \in \text{Imp}$ is a natural number, denoted $(\mathsf{C})^{\mathsf{b}} \in \mathbb{N}$, defined inductively as follows:

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

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

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

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

$$(\mathsf{C}_1 + \mathsf{C}_2)^{\mathsf{b}} \triangleq (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$$

$$(\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} \triangleq (\mathsf{C}_1)^{\mathsf{b}} + (\mathsf{C}_2)^{\mathsf{b}}$$

$$(\mathsf{C}^*)^{\mathsf{b}} \triangleq (|\mathit{vars}(\mathsf{C})| + 1)(\mathsf{C})^{\mathsf{b}}$$

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

In order to compute the bound of a program we introduce bounded environments, that we'll later use for lemma 4.1

Definition 4.3 (**Bound Environment**)**.** A bound environment (benv for short) is a total function $b : \mathit{Var} \to \mathbb{N}$. We define $\mathsf{bEnv} \triangleq \{b \mid b : \mathit{Var} \to \mathbb{N}\}$. Each command $\mathsf{C} \in \text{Imp}$ induces a benv transformer $[C]^{\mathsf{b}} : \mathsf{bEnv} \to \mathsf{bEnv}$, which is defined inductively as follows:

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

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

Lemma 4.1. *For all* $\mathsf{C} \in \mathit{Imp}$, $(\mathsf{C})^{\mathsf{b}} = \sum_{\mathtt{x} \in \mathit{vars}(\mathsf{C})} \left([\mathsf{C}]^{\mathsf{b}} b_0\right)(\mathtt{x})$, *with* $b_0 \triangleq \lambda x.0$.

Proof. For the following proof we work by induction on $\mathsf{C} \in \text{Imp}$:

Base cases:

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

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

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

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

Case $(\mathtt{x} := k)$ just notice that $(\mathtt{x} := k)^{\mathsf{b}} = k = \sum_{\mathtt{y} \in \mathit{vars}(\mathsf{C})} b_0[\mathtt{x} \mapsto b_0 + k] = [\mathtt{x} := k]^{\mathsf{b}} b_0$

Case $(\mathtt{x} := \mathtt{y} + k)$

Inductive cases:

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

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

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

Case (C^*)

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

□

4.3 Computing the analysis

intro

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

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

where:

(i). $\ell \geqslant 1$

(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 4.2 (**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 $m \triangleq \max\{|h_j| \mid j \in \{0,\ldots,\ell-1\}\} \in \mathbb{N}$ 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 $|p| = \ell \geqslant |X|$. Then, we show our claim by induction on $|p| = \ell \geqslant |X|$.
($|p| = |X|$): 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

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

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

Lemma 4.3. *Let* $\mathsf{C} \in$ *Imp.*
For all $\eta \in \mathbb{A}$ *and* $\mathtt{y} \in$ *Var, if* $\max([\![\mathsf{C}]\!]\eta\mathtt{y}) \neq \infty$ *and* $\max([\![\mathsf{C}]\!]\eta\mathtt{y}) > (\mathsf{C})^{\mathsf{b}}$ *then there exist a variable* $\mathtt{z} \in$ *Var and an integer* $h \in \mathbb{Z}$ *such that* $|h| \leqslant (\mathsf{C})^{\mathsf{b}}$ *and the following two properties hold:*

(i) $\max([\![\mathsf{C}]\!]\eta\mathtt{y}) = \max(\eta\mathtt{z}) + h$*;*

(ii) for all $\eta' \in \mathbb{A}$*, if* $\eta' \sqsupseteq \eta$ *then* $\max([\![\mathsf{C}]\!]\eta'\mathtt{y}) \geqslant \max(\eta'\mathtt{z}) + h$*.*

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

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

We distinguish two cases:

- If $\mathtt{y} \neq \mathtt{x}$, then for all $\eta' \in \mathbb{A}$ such that $\eta \sqsubseteq \eta'$ it holds $\bot \neq [\![\mathtt{x} \in S]\!]\eta' = \eta'[\mathtt{x} \mapsto \eta(\mathtt{x}) \sqcap \mathit{Int}(S)]$ 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 $\eta(\mathtt{x}) \in \mathit{Int}_*$ and
$$\max([\![\mathtt{x} \in S]\!]\eta\mathtt{y}) = \max(\eta(\mathtt{x}) \sqcap \mathit{Int}(S))$$
Note that it cannot be $\max(S) \in \mathbb{N}$. Otherwise, by Definition 4.2, $\max(\eta(\mathtt{x}) \sqcap \mathit{Int}(S)) \leqslant \max(S) = (\mathtt{x} \in S)^{\mathsf{b}}$, 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 $\max(\eta(\mathtt{x}) \sqcap \mathit{Int}(S)) = \max(\eta(\mathtt{x})) = \max(\eta(\mathtt{x})) + 0$. It is immediate to check that the same holds for all $\eta' \sqsupseteq \eta$, i.e.,
$$\max(\eta'(\mathtt{x}) \sqcap \mathit{Int}(S)) = \max(\eta'(\mathtt{x})) = \max(\eta'(\mathtt{x})) + 0$$
and thus the thesis follows with $\mathtt{z} = \mathtt{y} = \mathtt{x}$ and $h = 0$.

Case $(\mathtt{x} := k)$ 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 $\eta' \in \mathbb{A}$, $\eta \sqsubseteq \eta'$,

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

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 $\eta' \in \mathbb{A}$, $\eta \sqsubseteq \eta'$,

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

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 $\max(\mathsf{C}_2\eta_1\mathtt{y}) > (\mathsf{C}_1; \mathsf{C}_2)^{\mathsf{b}} \geqslant (\mathsf{C}_2)^{\mathsf{b}}$, 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 $\eta_1 \sqsubseteq \eta_1'$

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

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 $\eta \sqsubseteq \eta'$

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

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

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

and

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

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

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

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

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

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \bigsqcup_{i\in\mathbb{N}} \eta_i.$$

Since $[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} \neq \infty$, we have that for all $i \in \mathbb{N}$, $\eta_i\mathtt{y} \neq \infty$. Moreover, $\bigsqcup_{i\in\mathbb{N}} \eta_i$ 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 $i \geqslant m+1$

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y} = \eta_i\mathtt{y}.$$

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

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

(b) $\forall\eta' \sqsupseteq \eta_j.\, \max([\![\mathsf{C}+\mathsf{true}]\!]\eta'\mathtt{x}) \geqslant \max(\eta'\mathtt{z}) + h$.

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

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

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

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

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

$$\max([\![\mathsf{fix}(\mathsf{C})]\!]\eta\mathtt{y}) = \max(\eta\mathtt{z}) + h \tag{i}$$

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

$$\max([\![\mathsf{fix}(\mathsf{C})]\!]\eta'\mathtt{y}) \geqslant \max(\eta'\mathtt{z}) + h \tag{ii}$$

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

$$\sigma \triangleq (\mathtt{y}_0, 0) \rightarrow_{h_0} (\mathtt{y}_1, 1) \rightarrow_{h_1} \ldots \rightarrow_{h_m} (\mathtt{y}_{m+1}, m+1) \tag{4.1}$$

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

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

and we have that:

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

with $|h_j| \leqslant (\mathsf{C})^{\mathsf{b}}$ for $j \in \{i,\ldots,m\}$. Hence we can apply Lemma 4.2 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 $\eta' \sqsupseteq \eta_{k_1}$

$$\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{k_2-k_1} \eta' \mathtt{w}) \geqslant \max(\eta' \mathtt{w}) + h \tag{4.2}$$

This property (4.2) can be shown by induction on $k_2 - k_1 \geqslant 1$.
Then, an inductive argument allows us to show that for all $r \in \mathbb{N}$:

$$\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2-k_1)} \eta_{k_1} \mathtt{w}) \geqslant \max(\eta_{k_1} \mathtt{w}) + rh \tag{4.3}$$

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

$$\begin{array}{ll}
\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{(r+1)(k_2-k_1)} \eta_{k_1} \mathtt{w}) = & \\
\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{k_2-k_1} (\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2-k_1)} \eta_{k_1}) \mathtt{w}) \geqslant & \text{[by (4.2) as } \eta_{k_1} \sqsubseteq \llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2-k_1)} \eta_{k_1} \text{]} \\
\max(\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{r(k_2-k_1)} \eta_{k_1} \mathtt{w}) + h \geqslant & \text{[by inductive hypothesis]} \\
\max(\eta_{k_1} \mathtt{w}) + rh + h \geqslant \max(\eta_{k_1} \mathtt{w}) + (r+1)h &
\end{array}$$

However, This would contradict the hypothesis $\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \mathtt{y} \neq \infty$. In fact the inequality (4.3) 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 (4.1) 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 $\mu_{j+1} = \llbracket \mathsf{C} + \mathsf{true} \rrbracket \mu_j$, we have that $\max(\mu_{j+1} \mathtt{y}_{j+1}) \geqslant \max(\mu_{j+1} \mathtt{y}_j) + h_j$ and thus

$$\llbracket \mathsf{C} + \mathsf{true} \rrbracket^{m-k_1+1} \eta' \mathtt{y} = \mu_{m+1} \mathtt{y}_{m+1} \geqslant \max(\mathtt{y}_{k_1}) + \Sigma_{i=k_1}^m h_i = \max(\eta' \mathtt{w}) + \Sigma_{i=k_1}^m h_i$$

Since $\eta' = \llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta \sqsupseteq \eta_{k_1}$ we conclude

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

contradicting the assumption.
Therefore, the path σ of (4.1) must exist, and consequently

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

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

Let us now show (ii). Given $\eta' \sqsupseteq \eta$ from (4.1) we deduce that for all $j \in \{0, \ldots, m\}$, if we let $\mu_0 = \eta'$ and $\mu_{j+1} = \llbracket \mathsf{C} + \mathsf{true} \rrbracket \mu_j$, we have that

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

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

$$\max(\llbracket \mathsf{fix}(\mathsf{C}) \rrbracket \eta' \mathtt{y}) \geqslant \max(\mu_{m+1} \mathtt{y}) = \max(\mu_{m+1} \mathtt{y}_{m+1}) \geqslant \max(\eta' \mathtt{y}_0) + \Sigma_{i=0}^m h_i$$

as desired. □

Lemma 4.3 provides an effective algorithm for computing the interval semantics of commands. More precisely, given a command C, the corresponding finite set of variables $Var_{\mathsf{C}} \triangleq vars(\mathsf{C})$, and an interval environment $\rho : Var_{\mathsf{C}} \to Int$, we define

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

Then, when computing $[\![\mathsf{C}]\!]\rho$ on such ρ having a finite domain, we can restrict to a bounded interval domain $\mathbb{A}_{\mathsf{C},\rho} \triangleq (Var_{\mathsf{C}} \to Int_{\mathsf{C},\rho}) \cup \{\top, \bot\}$ where

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

Lemma 4.4. *Let $\mathsf{C} \in Imp$ be a command. Then, for all finitely supported $\rho : Var \to Int$, the abstract semantics $[\![\mathsf{C}]\!]\rho$ computed in $\mathbb{A}$ and in $\mathbb{A}_{\mathsf{C},\rho}$ coincide.*

Proof. What we want to prove is that given a command $\mathsf{C} \in$ Imp and an initial environment $\rho : Var \to Int$ the analysis in the lattice $\mathbb{A}$ and in the lattice $\mathbb{A}_{\mathsf{C},\rho}$ coincide. In other words what we want to prove is that for every variable $\mathtt{y}$ the values of $[\![\mathsf{C}]\!]\rho\mathtt{y}$ are in $\mathbb{A}_{\mathsf{C},\rho}$. We prove this by case analysis on each variable in Var_{C}.

For all $\mathtt{y} \in Var_{\mathsf{C}}$ we have two options:

- $\max([\![\mathsf{C}]\!]\rho\mathtt{y}) = \infty$, then the result of the analysis $[\![\mathsf{C}]\!]\rho\mathtt{y} = [a,\infty]$ for some $a \in \mathbb{N}$, which is in $Int_{\mathsf{C},\rho}$ by definition.
- $\max([\![\mathsf{C}]\!]\rho\mathtt{y}) \neq \infty$. In this case we have again 2 possibilities:
  - $\max([\![\mathsf{C}]\!]\rho\mathtt{y}) \leqslant (\mathsf{C})^{\mathsf{b}}$ and therefore by definition of $Int_{\mathsf{C},\rho}$ $[\![\mathsf{C}]\!]\rho\mathtt{y} \in Int_{\mathsf{C},\rho}$.
  - $\max([\![\mathsf{C}]\!]\rho\mathtt{y}) > (\mathsf{C})^{\mathsf{b}}$. In this case we can apply Lemma 4.3 and notice that

    $$\exists \mathtt{z} \in Var_{\mathsf{C}}, h \in \mathbb{Z} \mid |h| \leqslant (\mathsf{C})^{\mathsf{b}} \wedge \max([\![\mathsf{C}]\!]\rho\mathtt{y}) = \max(\rho\mathtt{z}) + h$$

    and we can conclude by noticing that $\max(\rho\mathtt{z}) + h \leqslant \max(\rho) + (\mathsf{C})^{\mathsf{b}}$, which means that $[\![\mathsf{C}]\!]\rho\mathtt{y} \in Int_{\mathsf{C},\rho}$.

Since for every given variable $\mathtt{y}$ the analysis $[\![\mathsf{C}]\!]\rho\mathtt{y}$ is in $Int_{\mathsf{C},\rho}$ then

$$[\![\mathsf{C}]\!]\rho \in \mathbb{A}_{\mathsf{C},\rho}$$

□

Bibliography

[Bel58] Richard Bellman. "On a routing problem". In: *Quarterly of Applied Mathematics* 16 (1958), pp. 87–90. DOI: 10.1090/qam/102435 (cit. on p. 1).

[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 (cit. on pp. iii, 1).

[Cut80] Nigel Cutland. *Computability: An introduction to recursive function theory*. Cambridge university press, 1980 (cit. on pp. 1, 13, 14).

[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 (cit. on p. 1).

[Hol06] G.J. Holzmann. "The power of 10: rules for developing safety-critical code". In: *Computer* 39.6 (2006), pp. 95–99. DOI: 10.1109/MC.2006.212 (cit. on pp. 23, 30).

[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 (cit. on p. 18).

[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 (cit. on p. 5).

[Min18] Antonie Miné. *Static Inference of Numeric Invariants by Abstract Interpretation*. Université Pierre et Marie Curie, Paris, France, 2018 (cit. on p. 2).

[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 (cit. on p. 16).

[Tar55] Alfred Tarski. "A lattice-theoretical fixpoint theorem and its applications." In: (1955) (cit. on p. 3).