University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master degree in Computer Science

Titolo della tesi

Supervisor
Prof. Prof 1

Co. Supervisor
Prof. Prof 2

Candidate
Luca Zaninotto

Academic Year 2023-2024

Abstract

Abstract

Acknowledgments

?

Contents

Chapter 1

Framework

1.1 The Imp language

We'll denote by $\mathbb{Z}$ the set of integers with the usual order plus two bonus elements $-\infty$ and $+\infty$, s.t. $-\infty \leqslant z \leqslant +\infty \quad \forall z \in \mathbb{Z}$. We also extend addition and subtraction by letting, for $z \in \mathbb{Z} \quad +\infty + z = +\infty - z = +\infty$ and $-\infty + z = -\infty - z = -\infty$.

We'll focus on the following non-deterministic language.

$$\begin{aligned} \text{Exp} \ni e ::= \quad & x \in S \mid x \in [a, b] \mid x \leqslant k \mid x > k \mid \texttt{true} \mid \texttt{false} \mid \\ & x := k \mid x := y + k \mid x := y - k \\ \text{Imp} \ni C ::= \quad & e \mid C + C \mid C; C \mid C* \end{aligned}$$

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

1.2 Semantics

In the following section we'll provide the semantics of the language we're working on and we'll make some observations; we'll also prove some properties of the language we'll use in the next sections.

The first building block is that of environments. We'll use environments to model the most precise invariant our semantic can describe for a program.

Definition 1.1 (Environments)**.** Environments are (total) maps from variables to (numerical) values

$$\text{Env} \triangleq \{\rho \mid \rho : \text{Var} \to \mathbb{Z}\}$$

Note: the set of notable variables is assumed to be finite.

Definition 1.2 (Semantics of Basic Expressions)**.** For basic expressions $e \in \text{Exp}$ the concrete

semantics $(\!|\cdot|\!) : \text{Exp} \to \text{Env} \to \text{Env} \cup \{\bot\}$ is recursively defined as follows:

$$
\begin{aligned}
(\!|x \in S|\!)\rho &\triangleq \begin{cases} \rho & \rho(x) \in S \\ \bot & \text{otherwise} \end{cases} \\
(\!|x \in [a,b]|\!)\rho &\triangleq \begin{cases} \rho & \rho(x) \in [a,b] \\ \bot & \text{otherwise} \end{cases} \\
(\!|x \leqslant k|\!)\rho &\triangleq \begin{cases} \rho & \rho(x) \leqslant k \\ \bot & \text{otherwise} \end{cases} \\
(\!|x > k|\!)\rho &\triangleq \begin{cases} \rho & \rho(x) > k \\ \bot & \text{otherwise} \end{cases} \\
(\!|\texttt{true}|\!)\rho &\triangleq \rho \\
(\!|\texttt{false}|\!)\rho &\triangleq \bot \\
(\!|x := k|\!)\rho &\triangleq \rho[x \mapsto k] \\
(\!|x := y + k|\!)\rho &\triangleq \rho[x \mapsto \rho(y) + k] \\
(\!|x := y - k|\!)\rho &\triangleq \rho[x \mapsto \rho(y) - k]
\end{aligned}
$$

The next building block is the concrete collecting semantics for the language, maps each program in Imp to a function on the $\mathbb{C}$ complete lattice.

Definition 1.3 (Concrete collecting domain)**.** The concrete collecting domain for the Imp language concrete collecting semantics is the complete lattice

$$\mathbb{C} \triangleq \langle 2^{\text{Env}}, \subseteq \rangle$$

We can therefore define the concrete collecting semantics for our language:

Definition 1.4 (Concrete collecting semantics)**.** The concrete collecting semantics for Imp is given by the total mapping

$$\langle \cdot \rangle : \text{Imp} \to \mathbb{C} \to \mathbb{C}$$

which maps each program $C \in \text{Imp}$ to its total mapping

$$\langle C \rangle : \mathbb{C} \to \mathbb{C}$$

on the complete lattice $\mathbb{C}$. The semantics is recursively defined as follows: given $X \in 2^{\text{Env}}$

$$
\begin{aligned}
\langle e \rangle X &\triangleq \{(\!|e|\!)\rho \mid \rho \in X, (\!|e|\!)\rho \neq \bot\} \\
\langle C_1 + C_2 \rangle X &\triangleq \langle C_1 \rangle X \cup \langle C_2 \rangle X \\
\langle C_1 ; C_2 \rangle X &\triangleq \langle C_2 \rangle (\langle C_1 \rangle X) \\
\langle C^* \rangle X &\triangleq \bigcup_{i \in \mathbb{N}} \langle C \rangle^i X
\end{aligned}
$$

Along with the collecting semantics we're also defining a one step transition relation. This will be useful to prove that finiteness is not decidable, as it helps to reason about program execution in a recursive way. The relation is defined on program states:

Definition 1.5 (Program State)**.** Program states are tuples of programs and program environments:

$$\text{State} \triangleq \text{Imp} \times \text{Env}$$

Definition 1.6 (Small step semantics)**.** The small step transition relation $\to$: State $\times$ (State$\cup$Env) is a small step semantics for the Imp language. It is defined based on the following rules

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

$$\frac{}{\langle C_1 + C_2, \rho\rangle \to \langle C_1, \rho\rangle}\ \text{sum}_1 \quad \frac{}{\langle C_1 + C_2, \rho\rangle \to \langle C_2, \rho\rangle}\ \text{sum}_2$$

$$\frac{\langle C_1, \rho\rangle \to \langle C_1', \rho'\rangle}{\langle C_1; C_2, \rho\rangle \to \langle C_1'; C_2, \rho'\rangle}\ \text{comp}_1 \quad \frac{\langle C_1, \rho\rangle \to \rho'}{\langle C_1; C_2, \rho\rangle \to \langle C_2, \rho'\rangle}\ \text{comp}_2$$

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

Lemma 1.1 (Collecting and small step link)**.** *For any $C \in Imp, X \in 2^{Env}$*

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

Therefore $\langle C\rangle X = \varnothing \iff \forall \rho \in X \langle C, \rho\rangle$ does not reach a final environment ρ_t.

Proof. by induction on C:

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

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

Inductive cases:

1. $C \equiv C_1 + C_2 : \langle C_1 + C_2\rangle X = \langle C_1\rangle X \cup \langle C_2\rangle X$, $\forall \rho \in X.\langle C_1 + C_2, \rho\rangle \to \langle C_1, \rho\rangle \vee \langle C_1 + C_2, \rho\rangle \to \langle C_2, \rho\rangle$ respectively according to rules sum_1 and sum_2. By inductive hypothesis

   $$\langle C_1\rangle X = \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_1, \rho\rangle \to^* \rho_t\} \quad \langle C_2\rangle X = \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_2, \rho\rangle \to^* \rho_t\}$$

   . Therefore

   $$\begin{aligned}
   \langle C_1 + C_2\rangle X &= \langle C_1\rangle X \cup \langle C_2\rangle X && \text{(by definition)}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_1, \rho\rangle \to^* \rho_t\} \cup \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_2, \rho\rangle \to^* \rho_t\} && \text{(by ind. hp)}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_1, \rho\rangle \to^* \rho_t \vee \langle C_2, \rho\rangle \to^* \rho_t\}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_1 + C_2, \rho\rangle \to^* \rho_t\}
   \end{aligned}$$

2. $C \equiv C_1; C_2 : \langle C_1; C_2\rangle X = \langle C_2\rangle(\langle C_1\rangle X)$. By inductive hp $\langle C_1\rangle X = \{\rho_t \in 2^{\text{Env}} \mid \rho \in X, \langle C_1, \rho\rangle \to^* \rho_t\} = Y$, by inductive hp again $\langle C_2\rangle Y = \{\rho_t \in 2^{\text{Env}} \mid \rho \in Y, \langle C_2, \rho\rangle \to^* \rho_t\}$. Therefore

   $$\begin{aligned}
   \langle C_1; C_2\rangle X &= \langle C_2\rangle(\langle C_1\rangle X) && \text{(by definition)}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho_x \in \{\rho_x \mid \rho \in X, \langle C_1, \rho\rangle \to^* \rho_x\}, \langle C_2, \rho_x\rangle \to^* \rho_t\} && \text{(by ind. hp)}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho \in X \langle C_1, \rho\rangle \to^* \rho_x \wedge \langle C_2, \rho_x\rangle \to^* \rho_t\} && \text{(by definition)}\\
   &= \{\rho_t \in 2^{\text{Env}} \mid \rho \in X.\langle C_1; C_2, \rho\rangle \to^* \rho_t\}
   \end{aligned}$$

3. $C \equiv C_1^*$ $\langle C^* \rangle X = \cup_{i \in \mathbb{N}} \langle C \rangle^i X$

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

Therefore we can notice that $\langle C \rangle X = \varnothing \iff \nexists \rho \in X \mid \langle C, \rho \rangle \to^* \rho_t$ for some $\rho_t \in \text{Env}$.

□

Chapter 2

Intervals

Interval semantics and analysis are among the most well known abstract interpretation standard abstract domains. 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.

2.1 Interval Analysis

The following chapter aims to introduce what we mean by interval analysis, giving some background and definition and finally proving the decidability of the analysis without the use of a widening operator. Notice that this is not in contrast to the more general results on concrete semantics, as interval analysis, falling under the broader umbrella of static analysis aims to provide a sound but maybe not complete analysis results.

Definition 2.1 (Interval domain)**.** We call by

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

the interval domain.

Definition 2.2 (Concretization map)**.** The concretization map $\gamma : Int \to 2^{\mathbb{Z}}$ is the following

$$\gamma([a,b]) = \{x \in \mathbb{Z} \mid a \leqslant x \leqslant b\}; \quad \gamma(\bot) = \varnothing$$

Definition 2.3 (Abstract domain)**.** The We'll call by $\mathbb{A}$ the abstract domain

$$\mathbb{A} \triangleq (Var \to int_*) \cup \{\bot\}$$

Definition 2.4 (Abstract concretization)**.** The abstract domain $\mathbb{A}$ concretization map is

$$\gamma(\bot) \triangleq \varnothing \tag{2.1}$$

$$\forall \eta \neq \bot, \quad \gamma(\eta) \triangleq \{\rho \in \text{Env} \mid \forall x \in \text{Var}.\rho(x) \in \gamma(\eta(x))\} \tag{2.2}$$

Chapter 3

Non relational collecting