University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master degree in Computer Science

Abstract Hoare logic

Supervisor

Prof. Francesco Ranzato

Co. Supervisor

Prof. Paolo Baldan

Candidate

Alessio Ferrarini

Academic Year 2023–2024

Abstract

In theoretical computer science . . .

Acknowledgments

To . . .

Contents

Chapter 1

Introduction and Background

In this chapter we give a brief introduction of program semantics

- Order theory
- Denotational semantics
- Abstract interpretation and Abstract Semantics
- Axiomatic semantics and Hoare Logic

1.1 Order theory

When defining the semantics of programming languages, the theory of *partially ordered sets* and *lattices* is fundamental. These concepts are at the core of denotational semantics [Sco70] and *Abstract Interpretation* [CC77], where the semantics of programming languages and abstract interpreters are defined as monotone functions over some complete lattice.

1.1.1 Partial Orders

Definition 1.1 (Partial order)**.** A partial order on a set X is a relation $\leq \subseteq X \times X$ such that the following properties hold:

- Reflexivity: $\forall x \in X,\ (x, x) \in \leq$
- Anti-symmetry: $\forall x, y \in X,\ (x, y) \in \leq \text{ and } (y, x) \in \leq \implies x = y$
- Transitivity: $\forall x, y, z \in X,\ (x, y) \in \leq \text{ and } (y, z) \in \leq \implies (x, z) \in \leq$

Given a partial order $\leq$, we will use $\geq$ to denote the converse relation $\{(y, x) \mid (x, y) \in \leq\}$ and $<$ to denote $\{(x, y) \mid (x, y) \in \leq \text{ and } x \neq y\}$.

From now on we will use the notation xRy to indicate $(x, y) \in R$.

Definition 1.2 (Partially ordered set)**.** A partially ordered set (or poset) is a pair $(X, \leq)$ in which $\leq$ is a partial order on X.

Definition 1.3 (Monotone function)**.** Given two ordered sets $(X, \leq)$ and $(Y, \sqsubseteq)$, a function $f : X \to Y$ is said to be monotone if $x \leq y \implies f(x) \sqsubseteq f(y)$.

Definition 1.4 (Galois connection)**.** Let $(C, \sqsubseteq)$ and $(A, \leq)$ be two partially ordered sets, a Galois connection written $\langle C, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq \rangle$, are a pair of functions: $\gamma : A \to D$ and $\alpha : D \to A$ such that:

- γ is monotone

- α is monotone
- $\forall c \in C\ c \sqsubseteq \gamma(\alpha(c))$
- $\forall a \in A\ a \leq \alpha(\gamma(a))$

Definition 1.5 (Fixpoint)**.** Given a function $f : X \to X$, a fixpoint of f is an element $x \in X$ such that $x = f(x)$.

We denote the set of all fixpoints of a function as $\text{fix}(f) = \{x \mid x \in X \text{ and } x = f(x)\}$.

Definition 1.6 (Least and Greatest fixpoints)**.** Given a function $f : X \to X$,

- We denote the *least fixpoint* as $\text{lfp}(f) = \min\ \text{fix}(f)$.
- We denote the *greatest fixpoint* as $\text{gfp}(f) = \max\ \text{fix}(f)$.

1.1.2 Lattices

Definition 1.7 (Meet-semilattice)**.** A poset $(X, \leq)$ is a meet-semilattice if $\forall x, y \in X, \exists z \in X$ such that $z = \inf\{x, y\}$, called the *meet*.

Usually, the meet of two elements $x, y \in X$ is written as $x \wedge y$.

Definition 1.8 (Join-semilattice)**.** A poset $(X, \leq)$ is a join-semilattice if $\forall x, y \in X, \exists z \in X$ such that $z = \sup\{x, y\}$, called the *join* or *least upper bound*.

Usually, the join of two elements $x, y \in X$ is written as $x \vee y$.

Observation 1.1. Both join and meet operations are idempotent, associative, and commutative.

Definition 1.9 (Lattice)**.** A poset $(X, \leq)$ is a lattice if it is both a join-semilattice and a meet-semilattice.

Definition 1.10 (Complete lattice)**.** A lattice $(X, \leq)$ is said to be complete if $\forall\ Y \subseteq X$:

- $\exists\ z \in X$ such that $z = \sup\ Y$
- $\exists\ z \in X$ such that $z = \inf\ Y$

We denote the *least element* or *bottom* as $\bot = \inf\ X$ and the *greatest element* or *top* as $\top = \sup\ X$.

Observation 1.2. A complete lattice cant be empty.

Definition 1.11 (Point-wise lift)**.** Given a complete lattice L and a set A we call *point-wise* lift of L the set of all functions $A \to L$ ordered point-wise $f \leq g \iff \forall a \in A\ f(a) \leq f(g)$.

Theorem 1.1 (Point-wise fixpoint)**.** *The leaft-fixpoint and greatest fixpoint on some point-wise lifted lattice on a monotone function defined point-wise is the point-wise lift of the function.*

$$\mathit{lfp}(\lambda p' a. f(p'(a))) = \lambda a. \mathit{lfp}(\lambda p'. f(a))$$
$$\mathit{gfp}(\lambda p' a. f(p'(a))) = \lambda a. \mathit{gfp}(\lambda p'. f(a))$$

Theorem 1.2 (Knaster-Tarski theorem)**.** *Let $(L, \leq)$ be a complete lattice and let $f : L \to L$ be a monotone function. Then $(\mathit{fix}(f), \leq)$ is also a complete lattice.*

Two direct consequences that both the greatest and the least fixpoint of f exists and are respectively $\top$ and $\bot$ of $\text{fix}(f)$.

Chapter 2

Framework

In this chapter we will introduce the general framework of *Abstract Hoare logic*

- The $\mathbb{L}$ programming language
- *Abstract inductive semantics*
- *Abstract Hoare logic*

2.1 The $\mathbb{L}$ programming language

2.1.1 Syntax

The $\mathbb{L}$ language is inspired by Dijkstra's guarded command languages [Dij74] but with the goal of beeing as general as possible by beeing parametric on a set of *base commands*. The $\mathbb{L}$ language is general enough to describe any imperative non deterministic programming language.

Definition 2.1 ($\mathbb{L}$ language syntax)**.** Given a set $Base$ of base commands, the set on valid $\mathbb{L}$ programs is defined by the following inductive definition:

$$
\begin{array}{rcllr}
\mathbb{L} & ::= & \mathbb{1} & & \text{Skip} \\
& | & b & & \text{Base command} \\
& | & C_1 \fatsemi C_2 & & \text{Program composition} \\
& | & C_1 + C_2 & & \text{Non deterministic choice} \\
& | & C^{\text{fix}} & & \text{Iteration}
\end{array}
$$

Where $C, C_1, C_2 \in \mathbb{L}$ and $b \in Base$.

Example 2.1. Usually the set of base commands contains a command $e?$ to discard execution that don't satisfy the predicate e and $x := y$ to assing the value y to the variable x.

2.1.2 Semantics

Fixed a set $\mathbb{S}$ of states (usually a collection of associations between variables names and values) and a family of partial functions $\llbracket \cdot \rrbracket_{base} : \mathbb{S} \hookrightarrow \mathbb{S}$ we can define the denotational semantics of programs in $\mathbb{L}$, the *collecting semantics* is a function $\llbracket \cdot \rrbracket : \mathbb{L} \to \wp(\mathbb{S}) \to \wp(\mathbb{S})$ that associates a program C and set of initial states to the set of states reached after executing the program C from the initial states.

Definition 2.2 ($\mathbb{L}$ denotational semantics)**.** Given a set $\mathbb{S}$ of states and a family of partial functions $[\![b]\!]_{base} : \mathbb{S} \hookrightarrow \mathbb{S}\ \forall b \in Base$ the denotational semantics is defined as follows:

$$\begin{aligned}
[\![\cdot]\!] &: \mathbb{L} \to \wp(\mathbb{S}) \to \wp(\mathbb{S}) \\
[\![\mathbb{1}]\!] &\stackrel{\text{def}}{=} id \\
[\![b]\!] &\stackrel{\text{def}}{=} \lambda P.\{[\![b]\!]_{base}(p) \downarrow \mid p \in P\} \\
[\![C_1 \fatsemi C_2]\!] &\stackrel{\text{def}}{=} [\![C_2]\!] \circ [\![C_1]\!] \\
[\![C_1 + C_2]\!] &\stackrel{\text{def}}{=} \lambda P.[\![C_1]\!]P \cup [\![C_2]\!]P \\
[\![C^{\text{fix}}]\!] &\stackrel{\text{def}}{=} \lambda P.\text{lfp}(\lambda P'.P \cup [\![C]\!]P')
\end{aligned}$$

Example 2.2. We can define the semantics of the base commands introduced in 2.1 as:

$$[\![e?]\!]_{base}(\sigma) \stackrel{\text{def}}{=} \begin{cases} \sigma & \sigma \models e \\ \uparrow & otherwise \end{cases}$$

$$[\![x := y]\!]_{base}(\sigma) \stackrel{\text{def}}{=} \sigma[x/eval(y, \sigma)]$$

Where *eval* is some evaluate function for the expressions on the left-hand side of assignments.

Theorem 2.1 (Complete lattice)**.** *$(\wp(\mathbb{S}), \subseteq)$ is a complete lattice.*

Proof. To prove that $(\wp(\mathbb{S}), \subseteq)$ is a complete lattice we exhibit: $\forall P \subseteq \wp(states)$

- $\inf P = \bigcap P$, it's clearly a lowerbound, and it's the greatest since any other set $Z \supsetneq \bigcap P$ contains some not in any of the elements in P.
- $\sup P = \bigcup P$, it's clearly an upper bound, and it's the smallest one since any other set $Z \subsetneq \bigcup P$ is missing some element that is in one of the elements of P.

□

Theorem 2.2 (Monotonicity)**.** *$\forall\, C \in \mathbb{L}$ $[\![C]\!]$ is monotone.*

Proof. We want to prove that $\forall P, Q \in \wp(\mathbb{S})$ and $C \in \mathbb{L}$

$$P \subseteq Q \implies [\![C]\!](P) \subseteq [\![C]\!](Q)$$

By structural induction on C:

- $\mathbb{1}$:

$$\begin{aligned}
[\![\mathbb{1}]\!](P) &= P && \text{By definition of } [\![\mathbb{1}]\!] \\
&\subseteq Q \\
&= [\![\mathbb{1}]\!](Q) && \text{By definition of } [\![\mathbb{1}]\!]
\end{aligned}$$

- b:

$$\begin{aligned}
[\![b]\!](P) &= \{[\![b]\!]_{base}(x) \downarrow \mid x \in P\} && \text{By definition of } [\![b]\!] \\
&\subseteq \{[\![b]\!]_{base}(x) \downarrow \mid x \in Q\} && \text{Since } P \subseteq Q \\
&= [\![b]\!](Q) && \text{By definition of } [\![b]\!]
\end{aligned}$$

- $C_1 \fatsemi C_2$:

  By inductive hypothesis $[\![C_1]\!]$ is monotone hence $[\![C_1]\!](P) \subseteq [\![C_2]\!](Q)$

$$\begin{aligned} [\![C_1 \fatsemi C_2]\!](P) &= [\![C_2]\!]([\![C_1]\!](P)) && \text{By definition of } [\![C_1 \fatsemi C_2]\!] \\ &\subseteq [\![C_2]\!]([\![C_1]\!](Q)) && \text{By inductive hypothesis on } [\![C_2]\!] \end{aligned}$$

- $C_1 + C_2$:

$$\begin{aligned} [\![C_1 + C_2]\!](P) &= [\![C_1]\!](P) \cup [\![C_2]\!](P) && \text{By definition of } [\![C_1 + C_2]\!] \\ &\subseteq [\![C_1]\!](Q) \cup [\![C_2]\!](P) && \text{By inductive hypothesis on } [\![C_1]\!] \\ &\subseteq [\![C_1]\!](Q) \cup [\![C_2]\!](Q) && \text{By inductive hypothesis on } [\![C_2]\!] \\ &= [\![C_1 + C_2]\!](Q) && \text{By definition of } [\![C_1 + C_2]\!] \end{aligned}$$

- C^{fix}:

$$\begin{aligned} [\![C^{\text{fix}}]\!](P) &= lfp(\lambda P'.P \cup [\![C]\!](P')) && \text{By definition of } [\![C^{\text{fix}}]\!] \\ &\subseteq lfp(\lambda P'.Q \cup [\![C]\!](P')) && \text{By theorem Kleene-Knaster-Tarski} \\ &= [\![C^{\text{fix}}]\!](Q) && \text{By definition of } [\![C^{\text{fix}}]\!] \end{aligned}$$

□

Lemma 2.1 ($[\![\cdot]\!]$ well-defined)**.** *$\forall\, C \in \mathbb{L}$ $[\![C]\!]$ is well-defined.*

Proof. From theorems 2.1, 2.2 and 1.2 all the least fixpoints in the definition of $[\![C^{\text{fix}}]\!]$ exists; for all the other commands the semantics is trivially well-defined. □

Observation 2.1. When the set of base commands contains a command to discard executions we can define the usual deterministic control flow commands as syntactic sugar.

$$if\ b\ then\ C_1\ else\ C_2 \overset{\text{def}}{=} (b? \fatsemi C_1) + (\neg b? \fatsemi C_2)$$

$$while\ b\ do\ C \overset{\text{def}}{=} (b? \fatsemi C)^{\text{fix}} \fatsemi \neg b?$$

Observation 2.2. Some other languages usually provide an iteration command usually denoted $C^\star$ whose semantics is $[\![C^\star]\!](P) \overset{\text{def}}{=} \bigcup_{n\in\mathbb{N}} [\![C]\!]^n(P)$, this is equivalent to C^{fix}, the reasoning on why a fixpoint formulation was chosen will become clear in 2.4.

2.2 Abstract inductive semantics

From the theory of abstract interpretation we know that the definition of the denotational semantics can be modified to work on any complete lattice as long that we can provide sensible function for the base commands. The rationale behind is the same as in the denotational semantics but instead representing collections of states with $\wp(\mathbb{S})$ now they are represented by an arbitrary complete lattice.

Definition 2.3 (Abstract inductive semantics)**.** Given a complete lattice A and a family of monotone functions $[\![b]\!]^{A}_{base} : A \to A$ $\forall b \in Base$ the abstract inductive semantics is defined as follows:

$$
\begin{aligned}
[\![\cdot]\!]^{A}_{ais} &: \mathbb{L} \to A \to A \\
[\![\mathbb{1}]\!]^{A}_{ais} &\overset{\text{def}}{=} id \\
[\![b]\!]^{A}_{ais} &\overset{\text{def}}{=} [\![b]\!]^{A}_{base} \\
[\![C_1 \fatsemi C_2]\!]^{A}_{ais} &\overset{\text{def}}{=} [\![C_2]\!]^{A}_{ais} \circ [\![C_1]\!]^{A}_{ais} \\
[\![C_1 + C_2]\!]^{A}_{ais} &\overset{\text{def}}{=} \lambda P. [\![C_1]\!]^{A}_{ais} P \vee_A [\![C_2]\!]^{A}_{ais} P \\
[\![C^{\text{fix}}]\!]^{A}_{ais} &\overset{\text{def}}{=} \lambda P. \text{lfp}(\lambda P'. P \vee_A [\![C]\!]^{A}_{ais} P')
\end{aligned}
$$

Ma è ben definito C^{fix}?, servirebbe la semantica monotona per avere l'esistenza di ogni lfp

From now on we will refer to the complete lattice used to define the abstract inductive semantics as *domain* borrowing the convention from abstract interpretation.

Observation 2.3. When picking as a domain the lattice $\wp(\mathbb{S})$ and as base commands $[\![b]\!]^{\wp(\mathbb{S})}_{base}(P) = \{[\![b]\!]_{base}(\sigma) \downarrow \mid \sigma \in P\}$ will result in obtaining the denotational semantics from the abstract inductive semantics. $\forall\, C \in \mathbb{L}$ $\forall P \in \wp(\mathbb{S})$

$$[\![C]\!]^{\wp(\mathbb{S})}_{ais}(P) = [\![C]\!](P)$$

Observation 2.4. There are some domains where $\exists\, C \in \mathbb{L}$ such that $\bigvee_{n \in \mathbb{N}}([\![C]\!]^{A}_{ais})^n(P) \neq \text{lfp}(\lambda P'. P \vee_A [\![C]\!]^{A}_{ais}(P'))$.

2.2.1 Galois connection semantics

Given a Galois connection $\langle C, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq \rangle$ if we have an abstract inductive semantics on domain C with base commands semantics $[\![b]\!]^{C}_{base}$ we can obtain another abstract inductive semantics on A by picking as base command semantics $[\![b]\!]^{A}_{base} = \alpha \circ [\![b]\!]^{C}_{base} \circ \gamma$.

Basterebbe soundess dei comandi base, più serve a qualcosa?

Theorem 2.3 (Soundness)**.** *The semantics* $[\![\cdot]\!]^{A}_{ais}$ *is sound with respect to* $[\![\cdot]\!]^{C}_{ais}$ *when* $[\![b]\!]^{A}_{base} = \alpha \circ [\![b]\!]^{C}_{base} \circ \gamma$.

$\forall\, R \in \mathbb{L}$ *and* $P \in C$

$$\alpha([\![R]\!]^{C}_{ais}(P)) \leq [\![R]\!]^{A}_{ais}(\gamma(P))$$

Bibliography

[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.

[Dij74] Edsger W. Dijkstra. "Guarded commands, non-determinacy and a calculus for the derivation of programs". see [**E**]WD:EWD472; circulated privately. June 1974. URL: http://www.cs.utexas.edu/users/EWD/ewd04xx/EWD418.PDF.

[Sco70] Dana Scott. *OUTLINE OF A MATHEMATICAL THEORY OF COMPUTATION*. Tech. rep. PRG02. OUCL, Nov. 1970, p. 30.