Roberto Bruni, Ugo Montanari

Models of Computation

– Monograph –

April 20, 2016

Springer

Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two facilities, which we may call intuition and ingenuity.

Alan Turing[1]

[1] The purpose of ordinal logics (from Systems of Logic Based on Ordinals), Proceedings of the London Mathematical Society, series 2, vol. 45, 1939.

Preface

The origins of this book lie their roots on more than 15 years of teaching a course on formal semantics to graduate Computer Science to students in Pisa, originally called *Fondamenti dell'Informatica: Semantica* (*Foundations of Computer Science: Semantics*) and covering models for imperative, functional and concurrent programming. It later evolved to *Tecniche di Specifica e Dimostrazione* (*Techniques for Specifications and Proofs*) and finally to the currently running *Models of Computation*, where additional material on probabilistic models is included.

The objective of this book, as well as of the above courses, is to present different *models of computation* and their basic *programming paradigms*, together with their mathematical descriptions, both *concrete* and *abstract*. Each model is accompanied by some relevant formal techniques for reasoning on it and for proving some properties.

To this aim, we follow a rigorous approach to the definition of the *syntax*, the *typing* discipline and the *semantics* of the paradigms we present, i.e., the way in which well-formed programs are written, ill-typed programs are discarded and the way in which the meaning of well-typed programs is unambiguously defined, respectively. In doing so, we focus on basic proof techniques and do not address more advanced topics in detail, for which classical references to the literature are given instead.

After the introductory material (Part I), where we fix some notation and present some basic concepts such as term signatures, proof systems with axioms and inference rules, Horn clauses, unification and goal-driven derivations, the book is divided in four main parts (Parts II-V), according to the different styles of the models we consider:

IMP:	imperative models, where we apply various incarnations of well-founded induction and introduce λ-notation and concepts like structural recursion, program equivalence, compositionality, completeness and correctness, and also complete partial orders, continuous functions, fixpoint theory;
HOFL:	higher-order functional models, where we study the role of type systems, the main concepts from domain theory and the distinction between lazy and eager evaluation;

CCS, π:	concurrent, non-deterministic and interactive models, where, starting from operational semantics based on labelled transition systems, we introduce the notions of bisimulation equivalences and observational congruences, and overview some approaches to name mobility, and temporal and modal logics system specifications;
PEPA:	probabilistic/stochastic models, where we exploit the theory of Markov chains and of probabilistic reactive and generative systems to address quantitative analysis of, possibly concurrent, systems.

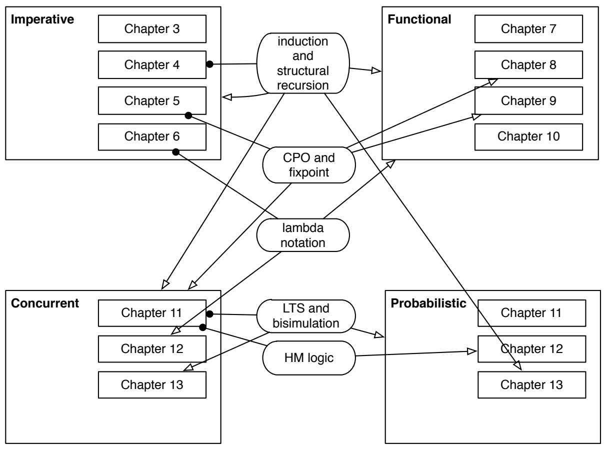
Each of the above models can be studied in separation from the others, but previous parts introduce a body of notions and techniques that are also applied and extended in later parts.

Parts I and II cover the essential, classic topics of a course on formal semantics.

Part III introduces some basic material on process algebraic models and temporal and modal logic for the specification and verification of concurrent and mobile systems. CCS is presented in good detail, while the theory of temporal and modal logic, as well as π-calculus, are just overviewed. The material in Part III can be used in conjunction with other textbooks, e.g., on model checking or π-calculus, in the context of a more advanced course on the formal modelling of distributed systems.

Part IV outlines the modelling of probabilistic and stochastic systems and their quantitative analysis with tools like PEPA. It poses the basis for a more advanced course on quantitative analysis of sequential and interleaving systems.

The diagram that highlights the main dependencies is represented below:



The diagram contains a squared box for each chapter / part and a rounded-corner box for each subject: a line with a filled-circle end joins a subject to the chapter where it is introduced, while a line with an arrow end links a subject to a chapter or part where it is used. In short:

Induction and recursion:	various principles of induction and the concept of structural recursion are introduced in Chapter 4 and used extensively in all subsequent chapters.

CPO and fixpoint:	the notion of complete partial order and fixpoint computation are first presented in Chapter 5. They provide the basis for defining the denotational semantics of IMP and HOFL. In the case of HOFL, a general theory of product and functional domains is also introduced (Chapter 8). The notion of fixpoint is also used to define a particular form of equivalence for concurrent and probabilistic systems, called bisimilarity, and to define the semantics of modal logic formulas.
Lambda-notation:	λ-notation is a useful syntax for managing anonymous functions. It is introduced in Chapter 6 and used extensively in Part III.
LTS and bisimulation:	Labelled transition systems are introduced in Chapter 11 to define the operational semantics of CCS in terms of the interactions performed. They are then extended to deal with name mobility in Chapter 13 and with probabilities in Part V. A bisimulation is a relation over the states of an LTS that is closed under the execution of transitions. The before mentioned bisimilarity is the coarsest bisimulation relation. Various forms of bisimulation are studied in Part IV and V.
HM-logic:	Hennessy-Milner logic is the logic counterpart of bisimilarity: two state are bisimilar if and only if they satisfy the same set of HM-logic formulas. In the context of probabilistic system, the approach is extended to Larsen-Skou logic in Chapter 15.

Each chapter of the book is concluded by a list of exercises that span over the main techniques introduced in that chapter. Solutions to selected exercises are collected at the end of the book.

Pisa, *Roberto Bruni*
February 2016 *Ugo Montanari*

Acknowledgements

We want to thank our friend and colleague Pierpaolo Degano for encouraging us to prepare this book and submit it to the EATCS monograph series. We thank Ronan Nugent and all the people at Springer for their editorial work. We acknowledge all the students of the course on *Models of Computation (MOD)* in Pisa for helping us to refine the presentation of the material in the book and to eliminate many typos and shortcomings from preliminary versions of this text. Last but not least, we thank Lorenzo Galeotti, Andrea Cimino, Lorenzo Muti, Gianmarco Saba, Marco Stronati, former students of the course on *Models of Computation*, who helped us with the LaTeX preparation of preliminary versions of this book, in the form of lecture notes.

Contents

Acronyms

$\sim$	operational equivalence in IMP (see Definition 3.3)
$\equiv_{den}$	denotational equivalence in HOFL (see Definition 10.4)
$\equiv_{op}$	operational equivalence in HOFL (see Definition 10.3)
$\simeq$	CCS strong bisimilarity (see Definition 11.5)
$\approx$	CCS weak bisimilarity (see Definition 11.16)
$\cong$	CCS weak observational congruence (see Section 11.8.2)
$\approx_d$	CCS dynamic bisimilarity (see Definition 11.17)
$\overset{\circ}{\sim}_E$	π-calculus early bisimilarity (see Definition 13.3)
$\overset{\circ}{\sim}_L$	π-calculus late bisimilarity (see Definition 13.4)
$\sim_E$	π-calculus strong early full bisimilarity (see Section 13.5.3)
$\sim_L$	π-calculus strong late full bisimilarity (see Section 13.5.3)
$\overset{\bullet}{\approx}_E$	π-calculus weak early bisimilarity (see Section 13.5.4)
$\overset{\bullet}{\approx}_L$	π-calculus weak late bisimilarity (see Section 13.5.4)
$\mathscr{A}$	interpretation function for the denotational semantics of IMP arithmetic expressions (see Section 6.2.1)
ack	Ackermann function (see Example 4.18)
Aexp	set of IMP arithmetic expressions (see Chapter 3)
$\mathscr{B}$	interpretation function for the denotational semantics of IMP boolean expressions (see Section 6.2.2)
Bexp	set of IMP boolean expressions (see Chapter 3)
$\mathbb{B}$	set of booleans
$\mathscr{C}$	interpretation function for the denotational semantics of IMP commands (see Section 6.2.3)
CCS	Calculus of Communicating Systems (see Chapter 11)
Com	set of IMP commands (see Chapter 3)
CPO	Complete Partial Order (see Definition 5.11)
$\text{CPO}_\perp$	Complete Partial Order with bottom (see Definition 5.12)
CSP	Communicating Sequential Processes (see Section 16.2)
CTL	Computation Tree Logic (see Section 12.1.2)
CTMC	Continuous Time Markov Chain (see Definition 14.15)

DTMC	Discrete Time Markov Chain (see Definition 14.14)
Env	set of HOFL environments (see Chapter 9)
fix	(least) fixpoint (see Definition 5.2.2)
FIX	(greatest) fixpoint
gcd	greatest common divisor
HML	Hennessy-Milner modal Logic (see Section 11.6)
HM-Logic	Hennessy-Milner modal Logic (see Section 11.6)
HOFL	A Higher-Order Functional Language (see Chapter 7)
IMP	A simple IMPerative language (see Chapter 3)
int	integer type in HOFL (see Definition 7.2)
Loc	set of locations (see Chapter 3)
LTL	Linear Temporal Logic (see Section 12.1.1)
LTS	Labelled Transition System (see Definition 11.2)
lub	least upper bound (see Definition 5.7)
$\mathbb{N}$	set of natural numbers
$\mathscr{P}$	set of closed CCS processes (see Definition 11.1)
PEPA	Performance Evaluation Process Algebra (see Chapter 16)
Pf	set of partial functions on natural numbers (see Example 5.13)
PI	set of partial injective functions on natural numbers (see Problem 5.12)
PO	Partial Order (see Definition 5.1)
PTS	Probabilistic Transition System (see Section 14.3.2)
$\mathbb{R}$	set of real numbers
$\mathscr{T}$	set of HOFL types (see Definition 7.2)
Tf	set of total functions from $\mathbb{N}$ to $\mathbb{N}_\perp$ (see Example 5.14)
Var	set of HOFL variables (see Chapter 7)
$\mathbb{Z}$	set of integers

Part III
HOFL: a higher-order functional language

This part focuses on models for sequential computations that are associated to HOFL, a higher-order declarative language that follows the functional style. Chapter 7 presents the syntax, typing and operational semantics of HOFL, while Chapter 9 defines its denotational semantics. The two are related in Chapter 10. Chapter 8 extends the theory presented in Chapter 5 to allow the definition of more complex domains, as needed by the type-constructors available in HOFL.

Chapter 10
Equivalence between HOFL denotational and operational semantics

Honest disagreement is often a good sign of progress. (Mahatma Gandhi)

Abstract In this chapter we address the correspondence between the operational semantics of HOFL from Chapter 7 and its denotational semantics from Chapter 9. The situation is not as straightforward as for IMP. A first discrepancy between the two semantics is that the operational one evaluates only closed (and typable) terms, while the denotational one can handle terms with variables, thanks to environments. Apart from this minor issue, the key fact is that the canonical forms arising from the operational semantics for the terms of type τ are more concrete than the mathematical elements of the corresponding domain $(V_\tau)_\perp$. Thus, it is inevitable that terms with different canonical forms can be assigned the same denotation. Vice versa, we show that terms with the same canonical form are always assigned the same denotation. Only for terms of type *int* we have a full agreement between the two semantics. On the positive side, a term converges operationally if and only if it converges denotationally. We conclude the chapter by discussing the equivalences over terms induced by the two semantics and by presenting an alternative denotational semantics, called *unlifted*, which is simpler but less expressive than the one studied in Chapter 9.

10.1 HOFL: Operational Semantics vs Denotational Semantics

As we have done for IMP, now we address the relation between the denotational and operational semantics of HOFL. One might expect to prove a complete equivalence, as in the case of IMP:

$$\forall t, c. \quad t \to c \quad \overset{?}{\Leftrightarrow} \quad \forall \rho.\ [\![t]\!]\rho = [\![c]\!]\rho$$

But, as we are going to show, the situation in the case of HOFL is more complex and the implication is valid in one direction only, i.e., the operational semantics is correct but not complete:

$$t \to c \Rightarrow \forall \rho.\ [\![t]\!]\rho = [\![c]\!]\rho \quad \text{but} \quad (\forall \rho.\ [\![t]\!]\rho = [\![c]\!]\rho) \not\Rightarrow t \to c$$

Let us consider a very simple example that shows the difference between the denotational and the operational semantics.

Example 10.1. Let $c_0 = \lambda x.\ x+0$ and $c_1 = \lambda x.\ x$ be two HOFL terms, where $x : int$. Clearly:

$$[\![c_0]\!]\,\rho = [\![c_1]\!]\,\rho \qquad \text{but} \qquad c_0 \not\to c_1$$

In fact, from the denotational semantics we get:

$$[\![c_0]\!]\,\rho = [\![\lambda x.\ x+0]\!]\,\rho = \lfloor \lambda d.\ d \underline{+}_{\perp} \lfloor 0 \rfloor \rfloor = \lfloor \lambda d.\ d \rfloor = [\![\lambda x.\ x]\!]\,\rho = [\![c_1]\!]\,\rho$$

but for the operational semantics we have that both $\lambda x.\ x$ and $\lambda x.\ x+0$ are already in canonical form and $c_0 \neq c_1$.

The counterexample shows that, at least for the functional type $int \to int$, there are different canonical forms with the same denotational semantics, namely terms which compute the same function in $[\mathbb{Z}_\perp \to \mathbb{Z}_\perp]_\perp$. One could think that a refined version of our operational semantics (e.g., one which could apply an axiom like $x+0=0$) would be able to identify exactly all the canonical forms which compute the same function. However this is not possible on computability grounds: since HOFL is able to compute all computable functions, the set of canonical terms which compute the same function is not recursively enumerable, while the set of theorems of every (finite) inference system is recursively enumerable.

Even if we cannot have a strong correspondence result between the operational and denotational semantics as it was the case for IMP, we can establish a full agreement between the two semantics w.r.t. the notion of termination. In particular, by letting the predicate $t\downarrow$ denote the fact that the term t can be reduced to some canonical form (called *operational convergence*) and $t \Downarrow$ denote the fact that the term $t : \tau$ is assigned a denotation other than $\perp_{(V_\tau)_\perp}$ (called *denotational convergence*), we have the perfect match:

$$t\downarrow \quad \Leftrightarrow \quad t\Downarrow$$

10.2 Correctness

We are ready to show the correctness of the operational semantics of HOFL w.r.t. the denotational one. Note that since the operational semantics is defined for closed terms only, the environment is inessential in the following theorem.

Theorem 10.1 (Correctness). *Let $t : \tau$ be a HOFL closed term and let $c : \tau$ be a canonical form. Then we have:*

$$t \to c \quad \Rightarrow \quad \forall \rho \in Env.\ [\![t]\!]\,\rho = [\![c]\!]\,\rho$$

Proof. We proceed by rule induction. So we prove

$$P(t \to c) \stackrel{\text{def}}{=} \forall \rho.\ [\![t]\!]\,\rho = [\![c]\!]\,\rho$$

for the conclusion $t \to c$ of each rule, when the predicate holds for the premises.

C_τ: The rule for terms in canonical forms (integers, pairs, abstraction) is

$$\frac{}{c \to c}$$

We have to prove $P(c \to c) \stackrel{\text{def}}{=} \forall \rho.\ [\![c]\!]\,\rho = [\![c]\!]\,\rho$, which is obviously true.

Arit.: Let us consider the rules for arithmetic operators $\mathrm{op} \in \{+,-,\times\}$:

$$\frac{t_1 \to n_1 \quad t_2 \to n_2}{t_1 \ \mathrm{op}\ t_2 \to n_1\ \underline{\mathrm{op}}\ n_2}$$

We assume the inductive hypotheses:

$$P(t_1 \to n_1) \stackrel{\text{def}}{=} \forall \rho.\ [\![t_1]\!]\,\rho = [\![n_1]\!]\,\rho = \lfloor n_1 \rfloor$$
$$P(t_2 \to n_2) \stackrel{\text{def}}{=} \forall \rho.\ [\![t_2]\!]\,\rho = [\![n_2]\!]\,\rho = \lfloor n_2 \rfloor$$

and we want to prove

$$P(t_1 \ \mathrm{op}\ t_2 \to n_1\ \underline{\mathrm{op}}\ n_2) \stackrel{\text{def}}{=} \forall \rho.\ [\![t_1 \ \mathrm{op}\ t_2]\!]\,\rho = \left[\!\!\left[n_1\ \underline{\mathrm{op}}\ n_2\right]\!\!\right]\rho.$$

We have:

$$\begin{array}{rll} [\![t_1 \ \mathrm{op}\ t_2]\!]\,\rho & = [\![t_1]\!]\,\rho\ \underline{\mathrm{op}}_\perp\ [\![t_2]\!]\,\rho & \text{(by definition of } [\![\cdot]\!]) \\ & = \lfloor n_1 \rfloor\ \underline{\mathrm{op}}_\perp\ \lfloor n_2 \rfloor & \text{(by inductive hypotheses)} \\ & = \lfloor n_1\ \underline{\mathrm{op}}\ n_2 \rfloor & \text{(by definition of } \underline{\mathrm{op}}_\perp) \\ & = \left[\!\!\left[n_1\ \underline{\mathrm{op}}\ n_2\right]\!\!\right]\rho & \text{(by definition of } [\![\cdot]\!]). \end{array}$$

Cond.: In the case of the conditional construct we have two rules to consider. For

$$\frac{t \to 0 \quad t_0 \to c_0}{\textbf{if}\ t\ \textbf{then}\ t_0\ \textbf{else}\ t_1 \to c_0}$$

we can assume

$$P(t \to 0) \stackrel{\text{def}}{=} \forall \rho.\ [\![t]\!]\,\rho = [\![0]\!]\,\rho = \lfloor 0 \rfloor$$
$$P(t_0 \to c_0) \stackrel{\text{def}}{=} \forall \rho.\ [\![t_0]\!]\,\rho = [\![c_0]\!]\,\rho$$

and we want to prove:

$$P(\textbf{if}\ t\ \textbf{then}\ t_0\ \textbf{else}\ t_1 \to c_0) \stackrel{\text{def}}{=} \forall \rho.\ [\![\textbf{if}\ t\ \textbf{then}\ t_0\ \textbf{else}\ t_1]\!]\,\rho = [\![c_0]\!]\,\rho$$

We have:

$$\begin{aligned}
[\![\mathbf{if}\ t\ \mathbf{then}\ t_0\ \mathbf{else}\ t_1]\!]\,\rho &= Cond([\![t]\!]\,\rho, [\![t_0]\!]\,\rho, [\![t_1]\!]\,\rho) && \text{(by def. of } [\![\cdot]\!]\text{)}\\
&= Cond(\lfloor 0 \rfloor, [\![t_0]\!]\,\rho, [\![t_1]\!]\,\rho) && \text{(by ind. hyp.)}\\
&= [\![t_0]\!]\,\rho && \text{(by def. of } Cond\text{)}\\
&= [\![c_0]\!]\,\rho && \text{(by ind. hyp.).}
\end{aligned}$$

The same procedure holds for the second rule of the conditional operator.

Proj.: Let us consider the rule for the first projection:

$$\frac{t \to (t_0, t_1) \quad t_0 \to c_0}{\mathbf{fst}(t) \to c_0}$$

We can assume

$$P(t \to (t_0,t_1)) \overset{\text{def}}{=} \forall \rho.\ [\![t]\!]\,\rho = [\![(t_0,t_1)]\!]\,\rho$$
$$P(t_0 \to c_0) \overset{\text{def}}{=} \forall \rho.\ [\![t_0]\!]\,\rho = [\![c_0]\!]\,\rho$$

and we want to prove

$$P(\mathbf{fst}(t) \to c_0) \overset{\text{def}}{=} \forall \rho.\ [\![\mathbf{fst}(t)]\!]\,\rho = [\![c_0]\!]\,\rho.$$

We have:

$$\begin{aligned}
[\![\mathbf{fst}(t)]\!]\,\rho &= \mathbf{let}\ v \Leftarrow [\![t]\!]\,\rho.\ \pi_1 v && \text{(by def. of } [\![\cdot]\!]\text{)}\\
&= \mathbf{let}\ v \Leftarrow [\![(t_0,t_1)]\!]\,\rho.\ \pi_1 v && \text{(by ind. hyp.)}\\
&= \mathbf{let}\ v \Leftarrow \lfloor([\![t_0]\!]\,\rho, [\![t_1]\!]\,\rho)\rfloor.\ \pi_1 v && \text{(by def. of } [\![\cdot]\!]\text{)}\\
&= \pi_1([\![t_0]\!]\,\rho, [\![t_1]\!]\,\rho) && \text{(by de-lifting)}\\
&= [\![t_0]\!]\,\rho && \text{(by def. of } \pi_1\text{)}\\
&= [\![c_0]\!]\,\rho && \text{(by ind. hyp.).}
\end{aligned}$$

The same procedure holds for the **snd** operator.

App.: The rule for application is:

$$\frac{t_1 \to \lambda x.\, t_1' \quad t_1'[^{t_0}/_x] \to c}{(t_1\ t_0) \to c}$$

We can assume:

$$P(t_1 \to \lambda x.\, t_1') \overset{\text{def}}{=} \forall \rho.\ [\![t_1]\!]\,\rho = [\![\lambda x.\, t_1']\!]\,\rho$$
$$P(t_1'[^{t_0}/_x] \to c) \overset{\text{def}}{=} \forall \rho.\ [\![t_1'[^{t_0}/_x]]\!]\,\rho = [\![c]\!]\,\rho$$

and we want to prove

$$P((t_1\ t_0) \to c) \overset{\text{def}}{=} \forall \rho.\ [\![(t_1\ t_0)]\!]\,\rho = [\![c]\!]\,\rho.$$

We have:

$$
\begin{aligned}
\llbracket (t_1\ t_0) \rrbracket \rho &= \mathbf{let}\ \varphi \Leftarrow \llbracket t_1 \rrbracket \rho .\ \varphi(\llbracket t_0 \rrbracket \rho) && \text{(by definition of } \llbracket\cdot\rrbracket)\\
&= \mathbf{let}\ \varphi \Leftarrow \llbracket \lambda x.\ t_1' \rrbracket \rho .\ \varphi(\llbracket t_0 \rrbracket \rho) && \text{(by ind. hypothesis)}\\
&= \mathbf{let}\ \varphi \Leftarrow \lfloor \lambda d.\ \llbracket t_1' \rrbracket \rho[^d/_x] \rfloor .\ \varphi(\llbracket t_0 \rrbracket \rho) && \text{(by definition of } \llbracket\cdot\rrbracket)\\
&= (\lambda d.\ \llbracket t_1' \rrbracket \rho[^d/_x])\ (\llbracket t_0 \rrbracket \rho) && \text{(by de-lifting)}\\
&= \llbracket t_1' \rrbracket \rho[^{\llbracket t_0 \rrbracket \rho}/_x] && \text{(by application)}\\
&= \llbracket t_1'[^{t_0}/_x] \rrbracket \rho && \text{(by Subst. Lemma)}\\
&= \llbracket c \rrbracket \rho && \text{(by ind. hypothesis).}
\end{aligned}
$$

Rec.: Finally, we consider the rule for recursion:

$$\frac{t[^{\mathbf{rec}\ x.\ t}/_x] \to c}{\mathbf{rec}\ x.\ t \to c}$$

We can assume:

$$P(t[^{\mathbf{rec}\ x.\ t}/_x] \to c) \stackrel{\text{def}}{=} \forall \rho.\ \llbracket t[^{\mathbf{rec}\ x.\ t}/_x] \rrbracket \rho = \llbracket c \rrbracket \rho$$

and we want to prove

$$P(\mathbf{rec}\ x.\ t \to c) \stackrel{\text{def}}{=} \forall \rho.\ \llbracket \mathbf{rec}\ x.\ t \rrbracket \rho = \llbracket c \rrbracket \rho.$$

We have:

$$
\begin{aligned}
\llbracket \mathbf{rec}\ x.\ t \rrbracket \rho &= \llbracket t \rrbracket \rho[^{\llbracket \mathbf{rec}\ x.\ t \rrbracket \rho}/_x] && \text{(by definition)}\\
&= \llbracket t[^{\mathbf{rec}\ x.\ t}/_x] \rrbracket \rho && \text{(by the Substitution Lemma)}\\
&= \llbracket c \rrbracket \rho && \text{(by inductive hypothesis)}
\end{aligned}
$$

Since there are no more rules to consider, we conclude the thesis holds. □

10.3 Equivalence (on Convergence)

Now we define the concept of convergence (i.e., termination) for the operational and the denotational semantics.

Definition 10.1 (Operational convergence). Let $t : \tau$ be a closed term of HOFL, we define the following predicate:

$$t \downarrow \quad \Leftrightarrow \quad \exists c \in C_\tau .\ t \longrightarrow c.$$

If $t\downarrow$, then we say that t *converges operationally*. We say that t *diverges*, written $t\uparrow$, if t does not converge operationally.

A term t converges operationally if the term can be evaluated to a canonical form c. For the denotational semantics we have that a term t converges if the evaluation function applied to t takes a value different from $\perp$.

Definition 10.2 (Denotational convergence). Let t be a closed term of HOFL with type τ, we define the following predicate:

$$t \Downarrow \quad \Leftrightarrow \quad \forall \rho \in Env, \exists v \in V_\tau.\ [\![t]\!]\,\rho = \lfloor v \rfloor .$$

If the predicate holds for t then we say that t *converges denotationally*.

We aim to prove that the two semantics agree at least on the notion of convergence. The implication $t \downarrow \Rightarrow\ t \Downarrow$ can be readily proved.

Theorem 10.2. *Let $t : \tau$ be a closed typable term of HOFL. Then we have:*

$$t \downarrow \quad \Rightarrow \quad t \Downarrow$$

Proof. If $t \to c$, then $\forall \rho.\ [\![t]\!]\,\rho = [\![c]\!]\,\rho$ by Theorem 10.1. But $[\![c]\!]\,\rho$ is a lifted value, (see Theorem 9.6) and thus it is different than $\perp_{(V_\tau)_\perp}$. □

Also the opposite implication $t \Downarrow \Rightarrow\ t \downarrow$ holds (for any closed and typable term t, see Theorem 10.3) but the proof is not straightforward: We cannot simply rely on structural induction; instead it is necessary to introduce a particular logical relation between elements of the interpretation domains and HOFL terms. We will only sketch the proof, but first we show that the standard structural induction does not help in proving the agreement of semantics about convergence.

Remark 10.1 (On the reason why structural induction fails for proving $t \Downarrow \Rightarrow\ t \downarrow$). The property $P(t) \stackrel{\text{def}}{=} t \Downarrow \Rightarrow\ t \downarrow$ cannot be proved by structural induction on t. Here we give some insights on the reason why it is so. Let us focus on the case of function application $(t_1\ t_0)$. By structural induction, we assume

$$P(t_1) \stackrel{\text{def}}{=} t_1 \Downarrow \Rightarrow\ t_1 \downarrow \qquad \text{and} \qquad P(t_0) \stackrel{\text{def}}{=} t_0 \Downarrow \Rightarrow\ t_0 \downarrow$$

and we want to prove $P(t_1\ t_0) \stackrel{\text{def}}{=} (t_1\ t_0) \Downarrow \Rightarrow\ (t_1\ t_0) \downarrow$.

Let us assume the premise $(t_1\ t_0) \Downarrow$ (i.e., $[\![(t_1\ t_0)]\!]\,\rho \neq \perp$) of the implication. We would like to prove that $(t_1\ t_0) \downarrow$, i.e., that $\exists c.\ (t_1\ t_0) \to c$. By definition of denotational semantics we have $t_1 \Downarrow$. In fact

$$[\![(t_1\ t_0)]\!]\,\rho \stackrel{\text{def}}{=} \textbf{let}\ \varphi \Leftarrow [\![t_1]\!]\,\rho.\ \varphi([\![t_0]\!]\,\rho)$$

and therefore $[\![(t_1\ t_0)]\!]\,\rho \neq \perp$ requires $[\![t_1]\!]\,\rho \neq \perp$. By the first inductive hypothesis we then have $t_1 \downarrow$ and by definition of the operational semantics it must be the case that $t_1 \to \lambda x.\ t_1'$ for some x and t_1'. By correctness (Theorem 10.1), we then have

$$[\![t_1]\!]\,\rho = [\![\lambda x.\ t_1']\!]\,\rho = \left\lfloor \lambda d.\ [\![t_1']\!]\,\rho[^d/_x] \right\rfloor .$$

Therefore:

$$\begin{aligned}
[\![(t_1\ t_0)]\!]\,\rho &= \textbf{let}\ \varphi \Leftarrow \left\lfloor \lambda d.\ [\![t_1']\!]\,\rho[^d/_x] \right\rfloor .\ \varphi([\![t_0]\!]\,\rho) && \text{(see above)}\\
&= (\lambda d.\ [\![t_1']\!]\,\rho[^d/_x])\ ([\![t_0]\!]\,\rho) && \text{(by de-lifting)}\\
&= [\![t_1']\!]\,\rho[^{[\![t_0]\!]\rho}/_x] && \text{(by functional application)}\\
&= [\![t_1'[^{t_0}/_x]]\!]\,\rho && \text{(by the Substitution Lemma)}
\end{aligned}$$

So $(t_1\ t_0) \Downarrow$ if and only if $t_1'[^{t_0}/_x] \Downarrow$. We would like to conclude by structural induction that $t_1'[^{t_0}/_x] \downarrow$ and then prove the theorem by using the rule:

$$\frac{t_1 \to \lambda x.\, t_1' \quad t_1'[^{t_0}/_x] \to c}{(t_1\ t_0) \to c}$$

but this is incorrect since $t_1'[^{t_0}/_x]$ is not a sub-term of $(t_1\ t_0)$ and we are not allowed to assume that $P(t_1'[^{t_0}/_x])$ holds.

Theorem 10.3. *For any closed typable term $t : \tau$ we have:*

$$t \Downarrow \quad \Rightarrow \quad t \downarrow$$

Proof. The proof exploits two suitable *logical relations*, indexed by HOFL types:

- $\precsim^c_\tau \subseteq V_\tau \times C_\tau$ that relates canonical forms to corresponding values in V_τ and that is defined by structural induction over types τ;
- $\precsim_\tau \subseteq (V_\tau)_\perp \times T_\tau$ that relates well-formed (closed) terms to values in $(V_\tau)_\perp$ and that is defined by letting:

$$d \precsim_\tau t \quad \stackrel{\text{def}}{=} \quad \forall v \in V_\tau.\ d = \lfloor v \rfloor \Rightarrow \exists c.\ t \to c \wedge v \precsim^c_\tau c$$

In particular, note that, by definition, we have $\perp_{(V_\tau)_\perp} \precsim_\tau t$ for any term $t : \tau$.

The logical relation on canonical forms is defined as follows

ground type: we simply let $n \precsim^c_{int} n$;
product type: we let $(d_0, d_1) \precsim^c_{\tau_0 * \tau_1} (t_0, t_1)$ iff $d_0 \precsim_{\tau_0} t_0$ and $d_1 \precsim_{\tau_1} t_1$;
function type: we let $\varphi \precsim^c_{\tau_0 \to \tau_1} \lambda x.\, t$ iff $\forall d_0 \in (V_{\tau_0})_\perp$ and $\forall t_0 : \tau_0$ closed, $d_0 \precsim_{\tau_0} t_0$ implies $\varphi(d_0) \precsim_{\tau_1} t[^{t_0}/_x]$.

Then one can show, by structural induction on $t : \tau$ that:

1. $\forall d, d' \in (V_\tau)_\perp.\ (d \sqsubseteq_{(V_\tau)_\perp} d' \wedge d' \precsim_\tau t) \Rightarrow d \precsim_\tau t$;
2. if $\{d_i\}_{i\in\mathbb{N}}$ is a chain in $(V_\tau)_\perp$ such that $\forall i \in \mathbb{N}.\ d_i \precsim_\tau t$, then $\bigsqcup_{i\in\mathbb{N}} d_i \precsim_\tau t$ (i.e., the predicate $\cdot \precsim_\tau t$ is inclusive).

Finally, by structural induction on terms, one can prove that $\forall t : \tau$ with $\mathrm{fv}(t) \subseteq \{x_1 : \tau_1, \ldots, x_k : \tau_k\}$, if $\forall i \in [1,k].\ d_i \precsim_{\tau_i} t_i$ then $[\![t]\!]\,\rho[^{d_1}/_{x_1}, \ldots, ^{d_k}/_{x_k}] \precsim_\tau t[^{t_1}/_{x_1}, \ldots, ^{t_k}/_{x_k}]$. In fact, taking $t : \tau$ closed, it follows from the definition of $\precsim_\tau$ that if $t \Downarrow$, i.e., $[\![t]\!]\,\rho = \lfloor v \rfloor$ for some $v \in V_\tau$, then $t \to c$ for some canonical form c, i.e., $t \downarrow$. □

10.4 Operational and Denotational Equivalences of Terms

In this chapter introduction we have shown that the denotational semantics is more abstract than the operational. In order to study the relationship between the operational and denotational semantics of HOFL we now introduce two equivalence relations between terms. Operationally two closed terms are equivalent if they both diverge or have the same canonical form.

Definition 10.3 (Operational equivalence). Let t_0 and t_1 be two well-typed terms of HOFL then we define a binary relation $\equiv_{op}$ as follows:

$$t_0 \equiv_{op} t_1 \quad \Leftrightarrow \quad (t_0 \uparrow \ \wedge t_1 \uparrow) \vee (\exists c.\ t_0 \to c \ \wedge \ t_1 \to c)$$

And we say that t_0 is operationally equivalent to t_1 if $t_0 \equiv_{op} t_1$.

We have also the denotational counterpart of the definition of equivalence.

Definition 10.4 (Denotational equivalence). Let t_0 and t_1 be two well-typed terms of HOFL then we define a binary relation $\equiv_{den}$ as follows:

$$t_0 \equiv_{den} t_1 \quad \Leftrightarrow \quad \forall \rho.\, [\![t_0]\!]\,\rho = [\![t_1]\!]\,\rho$$

And we say that t_0 is denotationally equivalent to t_1 if $t_0 \equiv_{den} t_1$.

Remark 10.2. Note that the definition of denotational equivalence terms applies also to non closed terms. Operational equivalence of non closed terms t and t' could also be defined by taking the closure of the equivalence w.r.t. the embedding of t and t' in any context $C[\cdot]$ such that $C[t]$ and $C[t']$ are also closed, i.e., by requiring that $C[t]$ and $C[t']$ are operationally equivalent for any context $C[\cdot]$.

From Theorem 10.1 it follows that: $\equiv_{op} \Rightarrow \equiv_{den}$.
As pointed out in Example 10.1: $\equiv_{den} \not\Rightarrow \equiv_{op}$.
So it is in this sense that we can say that the denotational semantics is *more abstract* then the operational one, because the former identifies more terms than the latter. Note that if we assume $t_0 \equiv_{den} t_1$ and $[\![t_0]\!]\,\rho \neq \bot$ then we can only conclude that $t_0 \to c_0$ and $t_1 \to c_1$ for some canonical forms c_0 and c_1. We have $[\![c_0]\!]\,\rho = [\![c_1]\!]\,\rho$, but nothing ensures that $c_0 = c_1$ (see Example 10.1 at the beginning of this chapter).

Only when we restrict our attention to the terms of HOFL that are typed as integers, then the corresponding operational and denotational semantics fully agree. This is because if c_0 and c_1 are canonical forms in C_{int} then it holds that $[\![c_0]\!]\,\rho = [\![c_1]\!]\,\rho \Leftrightarrow c_0 = c_1$. It can be proved *int* is the only type for which the full correspondence holds.

Theorem 10.4. *Let $t : int$ be a closed term of HOFL and $n \in \mathbb{N}$. Then:*

$$\forall \rho.\ [\![t]\!]\,\rho = \lfloor n \rfloor \quad \Leftrightarrow \quad t \to n$$

Proof. We prove the two implications separately.

$\Rightarrow$) If $[\![t]\!]\,\rho = \lfloor n \rfloor$, then $t \Downarrow$ and thus $t \downarrow$ by the soundness of denotational semantics, namely $\exists m$ such that $t \rightarrow m$, but then $[\![t]\!]\,\rho = \lfloor m \rfloor$ by Theorem 10.1, thus $n = m$ and $t \rightarrow n$.

$\Leftarrow$) Just Theorem 10.1, because $[\![n]\!]\,\rho = \lfloor n \rfloor$. $\square$

10.5 A Simpler Denotational Semantics

We conclude this chapter by presenting a simpler denotational semantics which we call *unlifted*, because it does not use the lifted domains. This semantics is simpler but also less expressive than the lifted one. We define the following new domains:

$$D_{int} \overset{\text{def}}{=} \mathbb{Z}_\perp \qquad D_{\tau_1 * \tau_2} \overset{\text{def}}{=} D_{\tau_1} \times D_{\tau_2} \qquad D_{\tau_1 \to \tau_2} \overset{\text{def}}{=} [D_{\tau_1} \to D_{\tau_2}]$$

Now we can let $Env' \overset{\text{def}}{=} Var \to \bigcup_\tau D_\tau$ and define the simpler interpretation function $[\![t : \tau]\!]' : Env' \to D_\tau$ as follows (where $\rho \in Env'$):

(exactly as before)

$$\begin{aligned}
[\![n]\!]'\rho &= \lfloor n \rfloor \\
[\![x]\!]'\rho &= \rho(x) \\
[\![t_1 \text{ op } t_2]\!]'\rho &= [\![t_1]\!]'\rho \ \underline{\text{op}}_\perp \ [\![t_2]\!]'\rho \\
[\![\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2]\!]'\rho &= Cond([\![t_0]\!]'\rho, [\![t_1]\!]'\rho, [\![t_2]\!]'\rho) \\
[\![\textbf{rec } x.\, t]\!]'\rho &= \text{fix}\, \lambda d.\ [\![t]\!]'\rho[^d/_x]
\end{aligned}$$

(updated definitions)

$$\begin{aligned}
[\![(t_1, t_2)]\!]'\rho &= ([\![t_1]\!]'\rho, [\![t_2]\!]'\rho) \\
[\![\textbf{fst}(t)]\!]'\rho &= \pi_1([\![t]\!]'\rho) \\
[\![\textbf{snd}(t)]\!]'\rho &= \pi_2([\![t]\!]'\rho) \\
[\![\lambda x.\, t]\!]'\rho &= \lambda d.\ [\![t]\!]'\rho[^d/_x] \\
[\![(t_1\ t_2)]\!]'\rho &= ([\![t_1]\!]'\rho)\ ([\![t_2]\!]'\rho)
\end{aligned}$$

Note that the “unlifted” semantics differ from the “lifted” one only in the cases of pairing, projections, abstraction and application. On the one hand the unlifted denotational semantics is much simpler to read than the lifted one. On the other hand the unlifted semantics is more abstract than the lifted one and does not express some interesting properties. For instance, consider the two HOFL terms:

$$t_1 \overset{\text{def}}{=} \textbf{rec } x.\, x \ :\ int \to int \qquad \text{and} \qquad t_2 \overset{\text{def}}{=} \lambda x.\ \textbf{rec } y.\, y \ :\ int \to int$$

In the lifted semantics we have $[\![t_1]\!]\,\rho = \perp_{[\mathbb{Z}_\perp \longrightarrow \mathbb{Z}_\perp]_\perp}$ and $[\![t_2]\!]\,\rho = \lfloor \perp_{[\mathbb{Z}_\perp \longrightarrow \mathbb{Z}_\perp]} \rfloor$, thus

$$t_1 \not\Downarrow \qquad \text{and} \qquad t_2 \Downarrow .$$

In the unlifted semantics $[\![t_1]\!]'\rho = [\![t_2]\!]'\rho = \perp_{[\mathbb{Z}_\perp \longrightarrow \mathbb{Z}_\perp]}$, thus

$$t_1 \not\Downarrow' \quad \text{and} \quad t_2 \not\Downarrow' .$$

Note however that $t_1 \uparrow$ while $t_2 \downarrow$, thus the property $t \downarrow \Rightarrow t \Downarrow'$ does not hold, at least for some $t : int \to int$, since $t_2 \downarrow$ but $t_2 \not\Downarrow'$. However, the property holds for the unlifted semantics in the case of integers.

As a concluding remark, we observe that the existence of two different, both reasonable, denotational semantics for HOFL shows that denotational semantics is, to some extent, an arbitrary construction, which depends on the properties one wants to express.

Problems

10.1. Prove that the HOFL terms:

$$t_1 \stackrel{\text{def}}{=} \textbf{rec } f.\ \lambda x.\ ((\lambda y.\ 1)\ (f\ x)) \qquad t_2 \stackrel{\text{def}}{=} \lambda x.\ 1$$

have the same type and the same denotational semantics but different canonical forms.

10.2. Let us consider the HOFL term

$$map \stackrel{\text{def}}{=} \lambda f.\ \lambda x.\ ((f\ \textbf{fst}(x)), (f\ \textbf{snd}(x)))$$

from Problem 7.5.

1. Write the denotational semantics of *map* and of $(map\ \lambda z.\ z)$.
2. Give two terms $t_1 : int$ and $t_2 : int$ such that the terms

$$((map\ \lambda z.\ z)(t_1,t_2)) \qquad ((map\ \lambda z.\ z)(t_2,t_1))$$

have different canonical forms but the same denotational semantics.

10.3. Consider the HOFL term

$$t \stackrel{\text{def}}{=} \textbf{rec } x.\ ((\lambda y.\ \textbf{if } y \textbf{ then } 0 \textbf{ else } 0)\ x).$$

from Problem 7.6. Compute its denotational semantics, checking the equivalence with its operational semantics.

10.4. Consider the HOFL term:

$$t \stackrel{\text{def}}{=} \textbf{rec } f.\ \lambda x.\ \textbf{if } \textbf{fst}(x) \times \textbf{snd}(x) \textbf{ then } x \textbf{ else } (f\ (f\ x)).$$

Derive the type, the canonical form and the denotational semantics of t. Finally show another term t' with the same denotational semantics as t but with different canonical form.

10.5. Consider the HOFL term:

$$t \overset{\text{def}}{=} \mathbf{rec}\ f.\ \lambda x.\ \mathbf{if}\ \mathbf{fst}(x) - \mathbf{snd}(x)\ \mathbf{then}\ x\ \mathbf{else}\ (f\ x).$$

Derive the type, the canonical form and the denotational semantics of t. Finally show another term t' with the same denotational semantics as t but with different canonical form.

10.6. Consider the HOFL term:

$$t \overset{\text{def}}{=} \mathbf{rec}\ F.\ \lambda f.\ \lambda n.\ \mathbf{if}\ (f\ n)\ \mathbf{then}\ 0\ \mathbf{else}\ ((F\ f)\ n).$$

Derive the type, the canonical form and the denotational semantics of t. Finally show another term t' with the same denotational semantics as t but with different canonical form.

10.7. Consider the HOFL term:

$$t \overset{\text{def}}{=} \mathbf{rec}\ f.\ \lambda x.\ \mathbf{if}\ (\mathbf{fst}(x)\ \mathbf{snd}(x))\ \mathbf{then}\ x\ \mathbf{else}\ (f\ x).$$

Derive the type, the canonical form and the denotational semantics of t. Finally show another term t' with the same denotational semantics as t but with different canonical form.

10.8. Modify the ordinary HOFL semantics by defining the denotational semantics of the conditional construct as follows

$$[\![\mathbf{if}\ t\ \mathbf{then}\ t_0\ \mathbf{else}\ t_1]\!]\,\rho = Condd([\![t]\!]\,\rho, [\![t_0]\!]\,\rho, [\![t_1]\!]\,\rho)$$

where

$$Condd(z, z_0, z_1) = \begin{cases} z_0 & \text{if } z = \lfloor 0 \rfloor\ \vee\ z_0 = z_1 \\ z_1 & \text{if } z = \lfloor n \rfloor\ \wedge\ n \neq 0 \\ \perp & \text{otherwise} \end{cases}$$

Assume that $t_0, t_1 : int$.

1. Prove that *Condd* is a monotonic, continuous function.
2. Show a HOFL term with a different semantics than the ordinary, and explain how the relation between operational and denotational semantics of HOFL is actually changed.

10.9. Modify the semantics of HOFL assuming the following operational semantics for the conditional command:

$$\frac{t_0 \to 0 \quad t_1 \to c_1 \quad t_2 \to c_2}{\mathbf{if}\ t_0\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2 \to\ c_1} \qquad \frac{t_0 \to n \quad n \neq 0 \quad t_1 \to c_1 \quad t_2 \to c_2}{\mathbf{if}\ t_0\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2 \to\ c_2}.$$

1. Exibit the corresponding denotational semantics.
2. Prove that also for the modified semantics it holds that $t \to c$ implies $[\![t]\!] = [\![c]\!]$.

3. Finally, compute the operational and the denotational semantics of $(fact\ 0)$, with

$$fact \stackrel{\text{def}}{=} \textbf{rec } f.\ \lambda x.\ \textbf{if } x \textbf{ then } 1 \textbf{ else } x \times (f\ (x-1))$$

and check if they coincide.

10.10. Suppose the operational semantics of projections is changed

$$\text{from} \quad \frac{t \to (t_1,t_2) \quad t_1 \to c}{\textbf{fst}(t) \to c} \qquad \text{to} \quad \frac{t \to (t_1,t_2) \quad t_1 \to c \quad t_2 \to c'}{\textbf{fst}(t) \to c}$$

and analogously for **snd**, without changing the denotational semantics.

1. Prove that the property $t \to c \Rightarrow [\![t]\!]\rho = [\![c]\!]\rho$ is still valid.
2. Exhibit a counterexample showing that the property $[\![t]\!]\rho \neq \bot \Rightarrow t \to c$ is no longer valid.
3. Finally, modify the denotational semantics to recover the above property and illustrate its validity for the counterexample previously proposed.

10.11. Modify the operational semantics of HOFL by taking the following rules for conditionals:

$$\frac{t \to 0 \quad t_0 \to c_0 \quad t_1 \to c_1}{\textbf{if } t \textbf{ then } t_0 \textbf{ else } t_1 \to c_0} \qquad \frac{t \to n \quad n \neq 0 \quad t_0 \to c_0 \ \ t_1 \to c_1}{\textbf{if } t \textbf{ then } t_0 \textbf{ else } t_1 \to c_1}.$$

without changing the denotational semantics. Prove that:

1. for any term t and canonical form c, we have $t \to c \ \Rightarrow\ \forall \rho.\ [\![t]\!]\rho = [\![c]\!]\rho$;
2. in general $t\Downarrow \not\Rightarrow t\downarrow$ (and exhibit a counterexample).

10.12. Suppose we extend HOFL with the inference rule:

$$\frac{t_1 \to 0}{t_1 \times t_2 \to 0}$$

as in Problem, 7.12.

1. Exhibit a counterexample showing that the property

$$\forall t, c. \quad t \to c \quad \Rightarrow \quad \forall \rho.\ [\![t]\!]\rho = [\![c]\!]\rho$$

is no longer valid.
2. Modify the denotational semantics so that the above correspondence is obtained, and prove that this is the case.
3. Repeat the exercise adding also the inference rule:

$$\frac{t_2 \to 0}{t_1 \times t_2 \to 0}.$$

10.13. Prove formally that

$$\text{if} \quad x \notin \text{fv}(t) \quad \text{then} \quad \mathbf{rec}\ x.\ t \text{ is equivalent to } t$$

employing both the operational and the denotational semantics.

10.14. Assume that the HOFL term t_0 has c_0 as canonical form.

1. Exploit the Substitution Lemma (Theorem 9.4) to prove that for every term t_1' we have
$$[\![t_1'[^{t_0}/_x]]\!]\,\rho \;=\; [\![t_1'[^{c_0}/_x]]\!]\,\rho.$$
2. Prove that if $t_1' : int$ and $\text{fv}(t_1') \subseteq \{x\}$, then $t_1'[^{t_0}/_x] \equiv_{op} t_1'[^{c_0}/_x]$.
3. Conclude that if we replace the lazy evaluation rule
$$\frac{t_1 \to \lambda x.\, t_1' \quad t_1'[t_0/x] \to c}{(t_1\ t_0) \to c}$$
with the eager rule
$$\frac{t_1 \to \lambda x.\, t_1' \quad t_0 \to c_0 \quad t_1'[c_0/x] \to c}{(t_1\ t_0) \to c}$$
then, if $(t_1\ t_0) \to c$ in the eager semantics, then $(t_1\ t_0) \to c$ in the lazy semantics.
4. Exhibit a simple counterexample such that $\exists c.\ (t_1\ t_0) \to c$ according to the lazy semantics but not to the eager one.
5. Finally, exhibit another counterexample where the type of t_1' is not int and the properties at points 2 and 3 do not hold.

10.15. Extend the operational semantics of HOFL to non-closed terms, by allowing canonical forms that are not closed but otherwise keeping the same inference rules. Show an example of reduction to canonical form for a non closed term t. Then, prove that the following properties are still valid:

1. subject reduction: $t : \tau$ and $t \to c$ implies $c : \tau$;
2. $t \to c$ implies $[\![t]\!]\,\rho = [\![c]\!]\,\rho$ (remind that the Substitution Lemma holds for any terms, also not closed ones);
3. $t \downarrow$ implies $t \Downarrow$;
4. $t_1 : int \to c_1$, $t_2 : int \to c_2$ and $[\![t_1]\!] = [\![t_2]\!]$ imply $c_1 = c_2$;
5. $t \to c$ implies $[\![t[^{\mathbf{rec}\ z.\ z}/_x]]\!]\,\rho = [\![c[^{\mathbf{rec}\ z.\ z}/_x]]\!]\,\rho$.
Hint: Exploit property 2 above and the Substitution Lemma.

10.16. Modify the denotational semantics of HOFL by restricting the use of the *lifting* domain construction only to integers, namely $V_{int} = \mathbb{Z}_\perp$ but $V_{\tau_1 * \tau_2} = V_{\tau_1} \times V_{\tau_2}$ and similarly for functions.

1. List all the modified clauses of the denotational semantics.
2. Prove that $t \to c$ implies $[\![t]\!]\,\rho = [\![c]\!]\,\rho$.
3. Finally, prove that it is not true that $t \to c$ implies $[\![t]\!]\,\rho \neq \perp$.
Hint: consider the HOFL term $t \overset{\text{def}}{=} \mathbf{rec}\ f.\ \lambda x.\ (f\ x)\ :\ int \to int$.