Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 322 (2016) 227–242

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Class of Reversible Primitive Recursive Functions

Luca Paolini[a,1,2], Mauro Piccolo[a,1,2] and Luca Roversi[a,1,2]

[a] *Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185, 10149 Torino*

Abstract

Reversible computing is bi-deterministic which means that its execution is both forward and backward deterministic, i.e. next/previous computational step is uniquely determined. Various approaches exist to catch its extensional or intensional aspects and properties. We present a class RPRF of reversible functions which holds at bay intensional aspects and emphasizes the extensional side of the reversible computation by following the style of Dedekind-Robinson Primitive Recursive Functions. The class RPRF is closed by inversion, can only express bijections on integers — not only natural numbers —, and it is expressive enough to simulate Primitive Recursive Functions, of course, in an effective way.

Keywords: Reversible computing, Recursive permutations, Primitive Recursive Functions.

1 Introduction

Reversible computing (sometimes called isentropic or adiabatic computing) is, on its own, an unconventional form of computing. Origins of reversible computing trace back to the study of entropy in physical systems [16]. The goal was relating thermodynamic properties of the system with the amount of information that it could carry around. In the sixties, Landauer was the first to define a technique for transforming irreversible computations into equivalent reversible ones [9]. Landauer thought his machines could not reversibly get rid of their undo trails. Lecerf first described a technique to uncompute histories [10], but he was unaware of the thermodynamic applications. Bennett [2] rediscovered Lecerf reversal. "Bennett's trick" corresponds to copying the output before uncomputing the undo trail, thereby showing for the first time reversible computations that could avoid entropy generation. The moral of these studies tells us that, if a physical system performs a logically irreversible

[1] Partially supported by the LINTEL project.

[2] Email: luca.paolini@unito.it, mauro.piccolo@unito.it,luca.roversi@unito.it

http://dx.doi.org/10.1016/j.entcs.2016.03.016

1571-0661/

operation then it must increase the entropy of the environment [19]. When a computational system erases a bit of information, it must dissipate $\ln 2 \times kT$ energy, where k is Boltzmann's constant and T is the temperature. For $T = 300$ Kelvins (room temperature), this is about 2.9×10^{-21} Joules (roughly, the kinetic energy of a single air molecule at room temperature). Today's computers erase a bit of information (in the above sense) every time they perform a logic operation, so their hunger for energy is ever-increasing. Reversible computing can avoid to use irreversible operations and entropy increasing.

Here above we have recalled the Physics related aspects that make reversible computation relevant. From a Computer Science foundational point of view reversible computing is interesting because it subsumes classical computing: every computation in a classical model can be simulated by a reversible one [14]. Moreover, aspects of reversible computation are ubiquitous in everyday classical computations. We can find them in activities spanning from software verification to programming languages, passing through computer architectures, as well as part of innovative computing models, like quantum, bio, chemical and molecular ones.

Reversible Turing-machines. Foundational studies on the notion of "reversible computation" exist. They have been chiefly devoted to frame the thermodynamic relations between entropy and computation via Turing-machines [1,2,6]. A reversible Turing-machine is both deterministic (like a classical Turing-machine) and backward-deterministic, i.e. it is bi-directionally deterministic. The backward determinism allows to easily reverse the computation, viz. we can undo a reversible program step by step eventually re-establishing former situations [1]. Only recently, recursion-theoretic arguments have been surveyed with some degree of systematization in [1].

This work develops a starting proposal to a recursion theory of reversible functions, in the line of Dedekind-Robinson-Kleene.

Dedekind-Robinson-Kleene Functions. We start recalling the distinguishing aspects of Kleene's Partial Recursive Functions [7], that we simply call Partial Recursive Functions, abbreviated as (RF). These functions form an extension of the Dedekind-Robinson Primitive Recursive Functions (PRF) this paper starts from.

Our starting point are RF and PRF for various reasons. First, we want to manage entities that compose because they stand for and are written as functions. Second, RF, as well as PRF, balance intensional and extensional aspects. Intensionally, they can be taken as programming languages whose semantics is given informally. Extensionally, RF deals with *partial functions* [3] while PRF with *total ones*, both shifting the focus on functions closer to what other computational models can express and providing support to functional, or compositional, programming.

[3] A relation between two sets A, B is a subset of the cartesian product $A \times B$. A relation is *functional* when $(a,b),(a,b') \in A \times B$ implies $b = b'$. A relation is *co-functional* when $(a,b),(a',b) \in A \times B$ implies $a = a'$. A relation is *total* whenever $a \in A$ implies that $b \in B$ exists such that $(a,b) \in A \times B$. A relation is *co-total* whenever $b \in B$ implies that $a \in A$ exists such that $(a,b) \in A \times B$. A function is a total functional relation. A partial function is a functional relation. A function is injective whenever its graph is a co-functional relation. A function is surjective whenever its graph is a co-total relation.