Theoretical Computer Science 564 (2015) 63–88

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Formal study of functional orbits in finite domains

Jean-François Dufourd

University of Strasbourg, CNRS, ICube, Pôle Technologique, Boulevard S. Brant, BP10413, 67412 Illkirch, France

A R T I C L E   I N F O

Article history:
Received 22 March 2013
Received in revised form 7 July 2014
Accepted 23 October 2014
Available online 28 October 2014
Communicated by D. Sannella

Keywords:
Formal specification
Functional orbits
Program correctness
Linked representation
Algebraic data type
Memory shape analysis
Computer-aided proof
Coq system

A B S T R A C T

In computer science, functional orbits – i.e., tracks left by the iterations of a function – in a finite domain naturally appear at a high or a low level. This paper introduces a Coq logical orbit framework, the purpose of which is to help rigorously developing software systems with some complex data structures from specification to implementation.

The result is a Coq library of orbit concepts formalized as definitions, lemmas and theorems. Most of them are inspired by our previous work in geometric modelling, where combinatorial hypermaps were used to describe surface subdivisions appearing in computational geometry problems, e.g., building convex hulls or Delaunay diagrams. Now, this domain remains a reference for us, but our results are drastically extended and usable in other computer science areas. The proof of Floyd's cycle-detection algorithm, known as "the tortoise and the hare", confirms that point.

The library contains operations to observe, traverse and update orbits – addition, deletion, mutation, transposition – with proofs of their behavior. It focuses on the important case where the involved function is a partial injection. In this case, it defines a connectivity relationship and evaluates the variation of the number of connected components during updates.

1. Introduction

We present a formal study of the (functional) orbit notion, its properties and operations, using the Coq proof assistant [26,4]. The goal of this work is to give a sound basis for a general concept which can help to correctly and fully specify and build software systems with some complex data structures.

Our background is a rich application domain consisting in the specification and implementation of computer libraries to interactively build and manipulate geometric objects [5,9,19,12,13,18,7]. Such libraries – e.g., CGAL [37], Topofil [5] or CGoGN [20] in the academic world, and CATIA [36] in the industrial world – are vital in design, architecture, animation, mechanics or robotics.

Everybody knows the orbit notion used in physics to model the gravitationally curved path during the motion of a planet or the wave-like behavior of an electron in an atom. In mathematics, the orbit of an element z denotes the set of positions which may be taken by the images of z under the action of a group G, the orbits forming a partition of G's support.

In this paper, the definition we retain is quite different, even if it has some similar consequences. Let X be any space, $f : X \to X$ be any total function defined on X, $D \subseteq X$ be any (sub)domain of X, and $z \in X$ be any element of X. Roughly speaking, the (first uninterrupted) track in D of the iterates $z_0 = z$, $z_1 = f\ z_0, \ldots, z_k = f\ z_{k-1}, \ldots$ is called the *f-orbit* of z in D.

E-mail address: jfd@unistra.fr.

http://dx.doi.org/10.1016/j.tcs.2014.10.041
0304-3975/© 2014 Elsevier B.V. All rights reserved.

It is unnecessary to recall the fundamental role of iterations in all mathematics and computer science areas. However, to our knowledge, the tracks left by them, i.e., the orbits, have never been studied as a universe of "first-class objects" which may be constructed or modified by operations like addition, deletion, mutation and transposition. For instance, in interactive geometric modelling, such orbit operations occur when manipulating high-level mathematical objects – e.g., space subdivisions, polygons and polyhedra – at the specification level, or low-level programming objects – e.g., singly- or doubly-linked lists representing geometric objects – at the implementation level.

This article only examines the case where D is *finite*, which is significant in computer science where the considered sets and iterations are often finite. Moreover, even if the starting point is clear, the correct and complete definition of orbits, operations and related notions is far from being immediate, especially because orbit operations can involve D or f, and because a tiny modification of one of them can have very large effects in the orbit universe.

The orbit notion was explicitly mentioned by several authors at conceptual level, for combinatoric aims (e.g., Berge [3]), for dynamic systems studies such as fractal investigations (e.g., Mendelbrot [32] and Barnsley-Demko [1]), and for geometric model studies (e.g., Tutte [38] and Lienhardt [38]). We have ourselves a rather good experience of formalization and use of orbits in geometric modelling [19,12–14,18,7]. However, we made an effort to extract this notion from our previous work and present it as a whole for new uses.

The orbit notion has been approached in different ways at a low level to model memory sharing and aliasing, mainly for proofs of programs with pointers using Hoare logic [25] and its extensions. A pioneering work introducing linked list segments, a non-repetition predicate, separated list systems, and rules to deal with them, is due to Burstall [8]. Initiated by Reynolds [34], *separation logic* is intensively studied for upgrading the problems of memory sharing at a logical level, and avoiding to prove that a modification in a part of the memory does not affect other parts. In fact, proofs involving various data structures – e.g., arrays, linear and cyclic data structures, possibly nested – can be achieved with our orbit results. A combinatorial map example is introduced hereafter with hierarchized data structures and a full hypermap application was developed in our recent work [16]. Classical sequence problems can also be formalized and solved using orbits. As a significant illustration, we show how the total correctness of Floyd's circuit-finding algorithm [28], also known as "the tortoise and the hare" algorithm, can be obtained from orbit features.

However, nobody seems to have considered the orbit notion as we want to do it. Our aim is to give rigorous definitions and to state and prove numerous properties, not only in (informal) mathematical style, but also in a formal language which can be handled interactively by some proof assistant. To be as generic as possible, our choice is to use an intuitionistic higher-level typed logic, or lambda-calculus, namely the Calculus of Inductive Constructions, to formalize our concepts, and the Coq system [26,4], which is an implementation of this calculus, to guide and support our interactive proofs. Nothing is added to the calculus, except the *axiom of extensionality*. This axiom states that two functions are equal if they are equal in every point. Finally, this paper also serves as introduction to a Coq library of specifications and proofs which can be reused each time the orbit notion is involved.

The rest of the paper is organized as follows. In Section 2, we mathematically define the notion of orbit and related concepts, and give some properties. In Section 3, we develop an example in geometric modelling where orbits are central at high and low levels. In Section 4, we formalize in Coq these notions and specialize them to prove static properties. In Section 5, we formalize and prove Floyd's circuit detection algorithm. In Section 6, we particularize the results in the important case where f is a partial injection. In Section 7, we present operations to add and remove an element of D, and their effects on the orbits. In Section 8, we study a mutation operation to change the image of an element of f, and its repercussion on the orbits. In Section 9, we examine a transposition operation to split or merge orbits which are circuits. In Section 10, we define a relation of connectivity for partial injections and study the effect of the previous operations on the number of connected components. We discuss related works in Section 11 and we conclude in Section 12.

The main features of the Coq language and system used in our specifications and proofs are reminded. The whole formalization process is described and explained, but the full details of the proofs are out of the scope of this article. However, the complete Coq (Version 8.4pl2) development is available [15].

2. Mathematical definitions and properties

In this section, we mathematically define and investigate the basic orbit notions and properties. They are formalized in Coq in the following, but we prefer to introduce them more intuitively with usual mathematical notations.

Let X be any space where the equality $=$ is decidable, $f : X \to X$ be any (*total*) function defined on X, $D \subseteq X$ be any *finite (sub)domain* of X, and $z \in X$ be *any element* of X. Let $z_k = f^k\, z$ be the k-th *iterate* of z by f, for $k \geq 0$ (with $z_0 = z$). We list sequences in brackets, [and], with [ ] for the empty sequence, and we write $In\, z\, l$, or $z \in l$ for readability, when z occurs in the sequence (or list) l of X's elements.

Since D is finite, during the iteration process from z, there is a time when the current iterate is outside D – possibly at the beginning – or encounters an iterate already met. This is reflected in the following definition:

Definition 1 *(Orbital sequence, length, orbit, limit, top).*

(i) The f-*orbital sequence* of z at the order $k \geq 0$, denoted by $orbs_f\, k\, z$, is [ ] if $k = 0$, and $[z_0, z_1, \ldots, z_{k-1}]$ otherwise.
(ii) The *length of the* f-*orbit* of z in D, denoted by $lorb_{f,D}\, z$, is the smallest integer p such that $z_p \notin D$ or $z_p \in orbs_f\, p\, z$.