Journal of Logical and Algebraic Methods in Programming 85 (2016) 824–846

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Factor theory and the unity of opposites

Roland Backhouse

School of Computer Science, University of Nottingham, Nottingham NG8 1BB, England, United Kingdom

A R T I C L E   I N F O

Article history:
Received 3 June 2015
Received in revised form 15 September 2015
Accepted 11 January 2016
Available online 29 January 2016

Keywords:
Galois connection
Regular algebra
Regular language
Factor matrix
Factor graph
Pattern matching

A B S T R A C T

The theory of factors of a regular language is used to illustrate the unity-of-opposites theorem of Galois connections. Left and right factors of a language are characterised as unions of right- and left-invariant equivalence classes, respectively, and this characterisation is exploited in the construction of the factor graph. The factor graph is a representation of the poset of left factors and, isomorphically by the unity of opposites, the poset of right factors. Two illustrative examples are given, one of which is the failure function used in the Knuth–Morris–Pratt pattern-matching algorithm.

Quien sabe por algebra sabe scientificamente. [Who knows by algebra knows scientifically.] Petrus Nonius Salaciensis (Pedro Nunes Salaciense), Libro de Algebra, 1567

1. Introduction

In 1971, I chanced upon a thin book entitled "Regular Algebra and Finite Machines" by J.H. Conway [7]. I was attracted by the word "Algebra" in its title and bought it without hesitation because I was convinced that algebra is the key to turning the art of programming into a science. It subsequently formed the basis of much of my PhD thesis [2].

Particularly inspiring for me was Conway's theory of what he called "factors" of a regular language and its application to the construction of "biregulators" and the derivation of regularity-preserving operations on languages. Compared to some other research papers that I had been wading through, Conway's book came as a breath of fresh air. A highlight was page 58 which contains a table of biregulators; in this table each line is an algebraic formula — in the form of a "biregulator" — representing a function mapping languages to languages. By applying the theorem that biregulators preserve regularity, he was thus able to prove that all the functions in the table map regular languages to regular languages. Here was evidence indeed of the power of algebra.

Disappointingly, shortly after completing my PhD the relevance of factor theory had not lived up to my expectations. My enthusiasm was reawakened when I spotted a connection between the "failure function" that is a vital element in the Knuth–Morris–Pratt pattern matching algorithm [9] and the "factor graph" of a regular language, which notion I had introduced in my thesis. With the help of an MSc student, Rudi Lutz, its connection with the KMP algorithm and its

E-mail addresses: rcb@cs.nott.ac.uk, roland@backhouse.org.

http://dx.doi.org/10.1016/j.jlamp.2016.01.003

generalisation to sets of patterns [1,15] was formulated and published in [4]. However the euphoria at this discovery proved short-lived; I was unable to find further practical applications and — based on the number of citations of Conway's book at the time — nor could others. (Although the book has become more widely cited as its author's fame has grown,[1] I still do not know of any publications other than my own or ones written by my colleagues that make use of and/or extend factor theory.)

It wasn't until the late 1980s that I began to look again at factor theory after learning about Galois connections, particularly in the context of relation algebra. I was inspired because I realised that Conway's "factors" could be formulated in terms of Galois connections and that many of his theorems were instances of more general properties of such connections. For my own benefit, I wrote a short technical note on the topic but have never attempted to publish it.

When asked to write an article to celebrate José Nuno Oliveira's 60th birthday, I knew immediately that I should write something about algebra and Galois connections: for the simple reason that I know that José loves both these topics. Since José already knows about basic factor theory, I looked again at my thesis to see whether there is anything "new" I could say about factors and Galois connections. And, indeed, there is! In my thesis, I characterised factors differently from Conway because this made it easier to formulate an algorithm to construct the factor graph of a language. On rereading the thesis, I realised that this alternative characterisation provides a non-trivial illustration of the "unity-of-opposites" theorem which Lambek and Scott [11] describe as "the most interesting consequence of a Galois correspondence". ("Unity of opposites" is the name I have given to the theorem; the name is not used by Lambek and Scott.) This then is what this paper is about.

To make the contents more accessible to other readers, the paper begins in section 2 with a short summary of the theory of Galois connections; this is followed in section 3 by a longer calculational presentation of Conway's factor theory that exploits their properties.

The introduction to Galois connections concludes with the unity-of-opposites theorem. Briefly, the theorem asserts an isomorphism between the partial orderings on the image sets of two Galois-connected functions. Its application is the subject of section 4. A summary of how the existence of factor graphs is established for regular languages is given in section 4.1. This is followed by an algorithm for its construction in section 4.2. It is at this point that we give a concrete illustration of the unity of opposites. Section 4.3 provides a further illustration based on the failure function used in the Knuth–Morris–Pratt pattern matching algorithm.

Apart from illustrating the unity-of-opposites theorem, several elements of the paper are novel in the sense that they have never been published in a journal or conference paper before now. This includes the calculational presentation of Conway's factor theory in section 3, the characterisation of factors as unions of certain (well-known) equivalence classes in section 3.3, and the algorithm to compute factor graphs in section 4.2.

2. Galois connections

2.1. Definition and examples

A Galois connection involves two partially ordered sets[2] $(A, \leq)$ and $(B, \preceq)$ and two functions, $F \in A \leftarrow B$ and $G \in B \leftarrow A$. These four components together form a *Galois connection* iff for all $x \in B$ and $y \in A$ the following holds[3]

$$F.x \leq y \equiv x \preceq G.y \quad .$$

This compact definition of a Galois connection was introduced in [14]. We refer to F as the *lower adjoint* and to G as the *upper adjoint*.

Since the context of the main contribution of this paper is language theory, it seems appropriate to use functions on languages as examples of Galois connections. Suppose T is a finite set. The elements of T are called *symbols* and T itself is called the *alphabet*. A *word* of *length* n, where n is a natural number, is a sequence of symbols of length n. The *empty word*, denoted by ε, is the word of length zero. *Concatenation* is the operation of forming a word of length $m+n$ from two words of lengths m and n by appending the latter after the former. For example, if $T = \{a,b\}$ then *ab* and *bba* are both words and their concatenation is the word *abbba*. We use T^* to denote the set of all words. A *language over alphabet* T is a subset of T^*.

Example 1. Below is a list of functions on languages each of which is a lower adjoint in a Galois connection. Each is followed by details of the Galois connection.

(a) The boolean-valued function $X \mapsto (w \in X)$ that determines whether a fixed word w is in the given language X.

[1] "Conway" is a common English name; "J.H. Conway" is the now-famous mathematician John Horton Conway.

[2] Galois connections can be defined for preordered sets but, for our purposes, we restrict the definition to posets.

[3] An infix dot is used to denote function application. The symbol "$\equiv$" denotes equality of booleans (regrettably, often introduced as "if and only if"). We do not use the standard untyped equality symbol here in order to avoid confusion with continued equalities and inequalities, e.g. $m < n = p \leq q$, which are conventionally read conjunctionally. We do use the conventional equality symbol for booleans in calculations when the meaning is indeed conjunctional, i.e. $p = q = r$ means $p = q$ ***and*** $q = r$ (and hence also $p = r$ by transitivity of equality) whatever the type of p, q and r.