International Journal of Approximate Reasoning 78 (2016) 103–115

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Advances in integrative statistics for logic programming

Nicos Angelopoulos [a,b], Samer Abdallah [c], Georgios Giamas [b]

[a] Cancer Genome Project, Welcome Trust Sanger Institute, Hinxton, CB10 1SA, UK
[b] Department of Surgery and Cancer, Division of Cancer, Imperial College, London, W12 0NN, UK
[c] Department of Computer Science, University College London, WC1E 6BT, UK

ARTICLE INFO

Article history:
Received 23 January 2016
Received in revised form 13 May 2016
Accepted 17 June 2016
Available online 5 July 2016

Keywords:
R statistical computing
Logic programming
Visualisation
Machine learning
Graph drawing
bioinformatics

ABSTRACT

We present recent developments on the syntax of *Real*, a library for interfacing two Prolog systems to the statistical language *R*. We focus on the changes in Prolog syntax within SWI-Prolog that accommodate greater syntactic integration, enhanced user experience and improved features for web-services. We recount the full syntax and functionality of *Real* as well as presenting a full application and sister packages which include Prolog code interfacing a number of common and useful tasks that can be delegated to *R*. We argue that *Real* is a powerful extension to logic programming, providing access to a popular statistical system that has complementary strengths in areas such as machine learning, statistical inference and visualisation. Furthermore, *Real* has a central role to play in the uptake of semantic web, computational biology and bioinformatics as application areas for research in logic programming.

1. Introduction

Real [4] is a low level interface between Prolog and *R* [14]. It enables the user to call *R* functions on Prolog data and communicate the results back to the logic system. The library works on two open source systems: YAP [6] and SWI-Prolog [25]. This is possible as YAP has a fairly complete emulation of SWI's *C* language interface [23]. Since its first introduction *Real* has evolved and has exerted some influence in advances to Prolog syntax. Furthermore, it has been used in a number of projects and in the process acquired a number of sister libraries. These libraries deliver Prolog predicates to useful tasks that can be best be dealt by existing *R* code. *Real* has thus be shown to be a useful and well integrated Prolog library that can provide access to the wealth of open source code available in *R*.

Here we focus on describing the full syntax of *Real 2* and its role in recent developments with syntactic changes in *SWI-7*. The changes in both systems have made the integration of *R* code into Prolog more natural and unobtrusive. Changes in the library itself had to be made to accommodate transition to the new Prolog syntax while preserving compatibility with traditional implementations. It is thus the case that the *Real* can be used in both of the supported Prolog systems, but only SWI-Prolog benefits from the new tighter integration.

R has a huge array of contributed code often accompanying published papers. It has particular strengths in statistical inference [19,8], machine learning [11,9] and data visualisation [20]. Within the specialist area of bioinformatics, Bioconductor [7] is a large agglomerating project that manages a large number of additional, user-contributed libraries.

Real gives access to *R* libraries that can complement Prolog's weaknesses in areas such as statistical inference and visualisation. With the library installed, it is straight forward with a basic grasp of *R* to call its functions on Prolog data. However,

E-mail address: nicos.angelopoulos@sanger.ac.uk (N. Angelopoulos).

http://dx.doi.org/10.1016/j.ijar.2016.06.008
0888-613X/

Table 1
Library's main predicates.

Indicator	Operator	Symbol	Description
r/2	<-	←	Evaluate *R* expression and assign result
r/1	<-	←	Evaluate *R* expression and ignore result
r_new/1	<<-	↞	Argument is a fresh *R* variable
↞ /2	<<-	↞	*r*/2 but with error if *R* variable exists
r_call/2	<-C++O	← ++	*r*/{1, 2} with options (O)
r_library/1			Load *R* library in a hookable manner
r_start/0			Start the connection to *R*
r_stop/0			Stop the connection to *R*
r_remove/1			Remove *R* variable
r_thread_loop/0			Start an *R* thread server
r_serve/0			Serve all *R* expressions on queue thread

for users with no prior exposure to *R* there still might be a barrier. To address this, and in order to increase general usability of the library a number of sister packages have been developed. We highlight some of the predicates that enable access to *R* code without any knowledge of *R*.

Central application areas since the inception of *Real* has been these of semantic web, bioinformatics and computational biology. In this paper we describe the role of *Real* in a web-based application as well as presenting sister libraries that here have evolved for addressing real world bioinformatics tasks in the context of a variety of projects: [27,12,16]. The main thesis of this paper is that Prolog can play a central role as a unifying platform in research in statistical and probabilistic areas such as web reasoning and bioinformatics, taking advantage of its strong grip on knowledge representation and reasoning and in combinations with recent advances with *Real* and web programming [24,10].

2. *Real*

In this section we describe the main features of *Real* and the innovations in the new version *Real 2*, which include: syntactic extensions that allow *R* code to be represented in a form that more closely resembles normal *R* syntax, the new predicate *r_library*/1, which provides a more flexible way to locate and load *R* libraries from their local filestore, and support for multiple Prolog threads to use a single *R* session, allowing *Real* to be used in SWI-Prolog's multithreaded web server framework. Taken together, these innovations allow a tighter and smoother integration of *R* code and enable Prolog programmers to tap in the wealth of statistical functions implemented in *R* with greater ease.

2.1. Real's predicates

Real 2 adopts the convention of a uniform prefix to all the library predicates. The full list of *Real*'s predicates along with the associated operators and brief descriptions is shown in Table 1. New additions include a hookable locator for *R* libraries, web server support, intuitive syntax for non-destructive assignment and a new interface predicate for mixing Prolog and *R* options with options for directing output to graphical devices.

With the new predicate *r_library*/1 users can load the standard *R* libraries in their local installation. In addition, the predicate can be directed to user specified locations where local, possibly, changed sources of such libraries can be preferentially loaded in *Real*. The flexibility allows for (a) specific code to be loaded only known to *Real* thus leaving the remainder of the *R* installation intact, and (b) user code that can be made available and can work either with the distributed version while having extra functionality when used with the altered sources.

2.2. Basic operation

The bulk of the interaction with *Real* is via a single predicate ← /2 which is also defined as an infix operator. It is similar to the Prolog *is*/2 operator, except that the term on the right-hand side is interpreted as an *R* expression and evaluated in the embedded *R* session. Within *Real*, ← /2 can be used to transfer data between *R* and Prolog, to apply *R* functions to Prolog data, retrieve *R* values as Prolog data, and destructively assigning values to *R* variables. Disambiguation clearly distinguishes the different modes, which can be summarised by:

$$-PlVar \leftarrow +Rexpr$$
$$+RAexpr \leftarrow +PlData$$
$$+RAexpr \leftarrow +Rexpr$$

Disambiguation of the call modes depends on whether the right hand side (RHS) is ground and if so on its term form. When the left hand side (LHS) of the operator is a free variable, the first mode is assumed, where the value of *Rexpr* is passed to *PlVar* after it has been evaluated in *R*. When the RHS is a *c*/*N* term or a list then the second mode is assumed and the Prolog data term in the RHS, *PlData*, is transferred to the assignable *R* expression in the LHS: *RAexpr*, typically an *R*