A Framework for Resource Dependent EDSLs in a Dependently Typed Language (Artifact)

Jan de Muijnck-Hughes
University of Glasgow, United Kingdom
Jan.deMuijnck-Hughes@glasgow.ac.uk

Edwin Brady
University of St Andrews, United Kingdom
ecb10@st-andrews.ac.uk

Wim Vanderbauwhede
University of Glasgow, United Kingdom
Wim.Vanderbauwhede@glasgow.ac.uk

Abstract

Idris' Effects library demonstrates how to embed resource dependent algebraic effect handlers into a dependently typed host language, providing run-time and compile-time based reasoning on type-level resources. Building upon this work, Resources is a framework for realising *Embedded Domain Specific Languages* (EDSLs) with type systems that contain domain specific substructural properties. Differing from Effects, Resources allows a language's substructural properties to be encoded within type-level resources that are associated with language variables. Such an association allows for multiple effect instances to be reasoned about autonomically and without explicit type-level declaration. Type-level predicates are used as proof that the language's substructural properties hold. Several exemplar EDSLs are presented that illustrates our framework's operation and how dependent types provide correctness-by-construction guarantees that substructural properties of written programs hold.

2012 ACM Subject Classification Software and its engineering → General programming languages; Software and its engineering → Language features; Software and its engineering → Domain specific languages; Software and its engineering → System modeling languages

Keywords and phrases Dependent Types, Algebraic Effect Handlers, Domain-Specific Languages, Embedded Domain Specific Languages, Idris, Substructural Type-Systems

Digital Object Identifier 10.4230/DARTS.6.2.2

Funding This work was funded by EPSRC projects: *Border Patrol: Improving Smart Device Security through Type-Aware Systems Design* (EP/N028201/1); and *Type-Driven Verification of Communicating Systems* – EP/N024222/1.

Related Article Jan de Muijnck-Hughes, Edwin Brady, and Wim Vanderbauwhede, "A Framework for Resource Dependent EDSLs in a Dependently Typed Language", in 34th European Conference on Object-Oriented Programming (ECOOP 2020), LIPIcs, Vol. 166, pp. 20:1–20:31, 2020.
https://doi.org/10.4230/LIPIcs.ECOOP.2020.20

Related Conference 34th European Conference on Object-Oriented Programming (ECOOP 2020), November 15–17, 2020, Berlin, Germany (Virtual Conference)

1 Scope

Resources is a general purpose framework for constructing Resource Dependent EDSLs that have domain specific substructural type-systems. We build on previous work in designing algebraic effect handlers in Idris [3, 2]. Rather than associating an effect's abstract resource with the program itself we associate it with a bound variable within the EDSL. Further, the list of possible effects is now constrained to an *a priori* set of domain specific effects. Such an association and

restriction leads to greater reasoning and manipulation of the effects within a Resource-Dependent EDSLs, thus enabling autonomic effect management and reasoning about the state of an effect's resource.

The framework has been realised within the dependently typed language Idris [1] and we present a collection of exemplar EDSLs that demonstrate how to use the framework. Specifically the exemplars are: *Files* which reasons about multiple concurrent File IO; *Wireless* which reasons about domain specific bigraph construction; and *Sessions* which captures value dependent global session descriptions.

Our artefact is the Idris implementation of Resources together with the paper's described case studies. We present the artefact within a Virtual Box[1] machine image that contains a working production environment. The machine image has been designed to work with virtual machine management tool Vagrant[2]. We have also provided the source code for Resources alongside the source code for inspection and construction outside the constraints imposed by the virtual machine setup.

2 Content

The artifact itself includes:

- `README.org`
  A listing of the archive's contents.
- `ecoop2020.box`
  A vagrant box containing the virtual machine that replicates an execution environment for the framework.
- `About-The-Artefact.pdf`
  A file containing instructions for accessing the artefacts, including software dependencies, together with more information detailing the artefacts relationship to the paper.
- `idris-doc`
  An offline copy of the Idris documentation to facilitate language understanding.
- `code/`
  The source files for the framework has been provided alongside the virtual machine to enable better viewing of the source. We provide a simple `Makefile` that will build the framework, and type-check the examples. The listing for `code` is:
  - `resources/`
    This directory contains the source code for Resources. It contains the source code for the framework and the examples in appropriate namespaces.
  - `src/`
    Contains the source code for the framework and the examples.
  - `resources_src_html/`
    A pretty-printed version of the source rendered as HTML files. The HTML highlighted source code was generated using an external program[3].
  - `resources_doc`
    Contains generated `idrisdoc`[4] documentation.

[1] `https://www.virtualbox.org`
[2] `https://www.vagrantup.com`
[3] `https://github.com/david-christiansen/idris-code-highlighter`
[4] `http://docs.idris-lang.org/en/v1.3.1/reference/documenting.html`

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the latest version of Resources has been made available at:

`https://github.com/border-patrol/resources`.

4 Tested platforms

The implementation of Resources was developed and tested on Ubuntu Mate LTS 18.04.2 with version 1.3.2 of Idris. We have packaged the artefact within a Virtual Box machine image managed by Vagrant to provide a reproducible production environment. This production environment comprises of Alpine Linux 3.11.5. Virtual Box and Vagrant are known to work across all the major platforms: Linux, macOS, and Windows. As an alternative, we have also provided the supporting source code. Idris is also known to work with all the major platforms. More information about the machine image, and working with the provided source code, is presented alongside the virtual machine image. We have not tested the framework on other platforms and see no reason for the framework and examples to not compile on any machine setup that supports Idris.

5 License

The artifact is available under *The Clear BSD License*.

6 MD5 sum of the artifact

9a75e4a5804bee0be7edfcf152ad8f4c

7 Size of the artifact

219M

References

1 Edwin Brady. Idris, a general-purpose dependently typed programming language: Design and implementation. *J. Funct. Program.*, 23(5):552–593, 2013. doi:10.1017/S095679681300018X.

2 Edwin Brady. Programming and reasoning with algebraic effects and dependent types. In Greg Morrisett and Tarmo Uustalu, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA – September 25 – 27, 2013*, pages 133–144. ACM, 2013. doi:10.1145/2500365.2500581.

3 Edwin Brady. Resource-dependent algebraic effects. In Jurriaan Hage and Jay McCarthy, editors, *Trends in Functional Programming - 15th International Symposium, TFP 2014, Soesterberg, The Netherlands, May 26-28, 2014. Revised Selected Papers*, volume 8843 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2014. doi:10.1007/978-3-319-14675-1_2.