Chapter 1

Literature Review

Objective: Design and implement a type-safe eDSL (embedded domain specific language) and a framework for declarative spreadsheet generation in Haskell.

1.1 Chapter overview

This chapter presents the literature review process and a survey of previous works on eDSL design and functional programming approaches to spreadsheets generation. Section 1.2 lists the literature review questions. Next, Section 1.3 summarizes the initial literature search process and its results. Section 1.4 presents a set of relevance criteria. Finally, Section 1.5 overviews the answers to the literature review questions presented in Section 1.2.

1.2 Literature review questions

The literature review aims at answering the following questions:

1. What were the previous attempts at type-safe spreadsheet generation via Haskell?
2. Which Haskell features were used in the existing eDSLs?

3. Which DSL design techniques were used in such eDSLs?

1.3 Search engines and queries

To begin with, the following search engines were used: Semantic Scholar [1], Hackage [2], HaskellWiki [3], Google [4], and GitHub [5]. These platforms were selected as all of them provided short annotations of stored resources. Later on, such annotations accelerated the preliminary selection of relevant articles and projects. Additionally, different sets of queries were used on each platform depending on: 1) a platform's search mechanism; 2) numbers of on-topic results obtained via other queries on that platform. Table I demonstrates the search engines, queries, and the numbers of preliminary selected search results, excluding duplicates.

TABLE I
Search results

Search engine	**Search queries**	**Results**
Semantic Scholar	haskell embedded domain specific language	25
	spreadsheet functional programming	11
	haskell edsl	7
	spreadsheet generation dsl	6
	spreadsheet functional language	2
	functional excel	2
Google Scholar	spreadsheet generation	3
	functional excel	2
	spreadsheet dsl	2
	spreadsheet functional programming	1
Hackage	languages	755

Search engine	Search queries	Results
	sheet	4
HaskellWiki	Research papers/Domain specific languages	48
	Embedded domain specific language	6
GitHub	excel language:Haskell	7
	spreadsheet language:Haskell	7
YouTube	Lambdaconf DSL	2

1.4 Relevance criteria

It was decided that each relevant work should:

1) Be published 1999 or later. A significant number of papers on eDSLs were published between the publications of *Haskell 98* and *Haskell 2010* standards;
2) Be written in English;
3) Show Haskell implementation source code or contain a link to such code;
4) Desirably, explain how DSL design techniques were implemented in Haskell.
5) Desirably, demonstrate a way to model or generate spreadsheets via Haskell;

1.5 Selected literature overview

1.5.1 Works on spreadsheet generation

There exist several works on spreadsheet generation via Haskell. Wakeling [6] made a Haskell backend for Microsoft Excel. His approach was to write Haskell functions in a separate file, enter them as comments into a spreadsheet, and make a Haskell interpreter HUGS convert them into numeric values. The author admitted that the shortcomings of this interface were slowness of calculations and lack of error reporting. Additionally, there was no way to write formulas in a spreadsheet-native expression language. Later, Zaborsky created a Haskell library Xlsx [7] for reading, imperative construction, and writing of spreadsheets in Office Open XML xlsx format. The weakness of this library is that it did not provide a DSL to express formulas in a form other than plain text. Next, Thomasson [8] made a simple DSL for spreadsheets generation. The main drawbacks of his program were that its outputs were not suitable for existing spreadsheet systems. Also, the author used fixed cell references which made spreadsheet layouts inflexible. One of the later works was the ComonadSheet [9] library. It featured support for relative and absolute cell references and infinite cell streams. Nevertheless, it shared the problem of fixed cell references with Thomasson's work. Further, Kudasov [10] suggested the notion of tables and their monadic composition. This approach allows to represent tables and their headers as trees and provide a Haskell eDSL for a spreadsheet description. Although the author did not elaborate on cell reference tracking and formulas expression, his ideas lay the foundation of this thesis.

1.5.2 Haskell features in existing eDSLs

TABLE II
Haskell features in existing eDSLs

Feature	eDSL paper
Template Haskell	[11], [12], [13], [14], [15]
Monads	[11], [14], [16], [17], [18]
Type families	[17], [19]
Custom operators	[20]

1.5.3 DSL design techniques

Bernauer and Eisenberg [11] use runtime instance lookup for data deserialization. This is achieved by collecting all class instances of a type into a monoid and putting them into a map indexed with type names. Also, the authors claim that it is permittable in their case to violate the monad laws, though a more correct approach would be to use the recent `ApplicativeDo` GHC extension. They argue that this extension will provide users with convenient `do`-notaion and increase the implementation efficiency.

Ekblad [17] suggests using closed type families. This allows for writing high-level code in a Haskell eDSL and guarantees type-safe conversion into an intermediate language. Furthermore, he notes that to make generated code efficient it is necessary to make inefficient constructs irrepresentable in an eDSL. Another possible approach mentioned in the article is function inlining. However, this approach may lead to a problem of having too many local variables at once.

Evans et al.[19] mention Indexed Monadic Catamorphism, a technique for recursion elimination from foldings of datatypes. To make catamorphisms modular, they further encode folding algebra as a type class. Also, they introduce custom operators to make the code more readable.

1.6 Conclusion

Overall, there were not discovered any works that fully addressed the problem of type-safe declarative spreadsheet generation using Haskell. That is why, this thesis will build upon the work by Kudasov [10] and apply the DSL design techniques mentioned in Section 1.5.3.

Bibliography cited

[1] "Semantic scholar | AI-powered research tool." (), [Online]. Available: https://www.semanticscholar.org/ (visited on 10/25/2022).

[2] "Packages by category | hackage." (), [Online]. Available: https://hackage.haskell.org/packages/ (visited on 10/25/2022).

[3] "HaskellWiki." (), [Online]. Available: https://wiki.haskell.org/Haskell (visited on 10/25/2022).

[4] "Google." (), [Online]. Available: https://www.google.com/?hl=en (visited on 10/25/2022).

[5] "Build software better, together," GitHub. (), [Online]. Available: https://github.com (visited on 10/25/2022).

[6] D. Wakeling, "Spreadsheet functional programming," *J. Funct. Prog.*, vol. 17, no. 1, pp. 131–143, Jan. 2007, ISSN: 0956-7968, 1469-7653. DOI: 10.1017/S0956796806006186. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0956796806006186/type/journal_article (visited on 10/22/2022).

[7] "Zotero | groups > thesis-spreadsheet." (), [Online]. Available: https://www.zotero.org/groups/4772792/thesis-spreadsheet (visited on 10/25/2022).

[8] "Zotero | groups > thesis-spreadsheet." (), [Online]. Available: https://www.zotero.org/groups/4772792/thesis-spreadsheet (visited on 10/25/2022).

[9] "Zotero | groups > thesis-spreadsheet." (), [Online]. Available: https://www.zotero.org/groups/4772792/thesis-spreadsheet (visited on 10/25/2022).

[10] "Zotero | groups > thesis-spreadsheet." (), [Online]. Available: https://www.zotero.org/groups/4772792/thesis-spreadsheet (visited on 10/25/2022).

[11] A. Bernauer and R. Eisenberg, "Eiger: Auditable, executable, flexible legal regulations," *undefined*, 2022. DOI: 10.48550/arXiv.2209.04939. [Online]. Available: https://www.semanticscholar.org/reader/3532b1cd57aa2972f0bd7dd09698e98c4cccbdd0 (visited on 10/22/2022).

[12] J. García-Garland, A. Pardo, and M. Viera, "Attribute grammars fly first-class... safer!: Dealing with DSL errors in type-level programming," in *Proceedings of the 31st Symposium on Implementation and Application of Functional Languages*, Singapore Singapore: ACM, Sep. 25, 2019, pp. 1–12, ISBN: 978-1-4503-7562-7. DOI: 10.1145/3412932.3412942. [Online]. Available: https://dl.acm.org/doi/10.1145/3412932.3412942 (visited on 10/22/2022).

[13] J. Bedő, “BioShake: A haskell EDSL for bioinformatics workflows,” *PeerJ*, vol. 7, 2019. DOI: 10.7717/peerj.7223. [Online]. Available: https://www.readcube.com/articles/10.7717%2Fpeerj.7223 (visited on 10/22/2022).

[14] M. Viera, F. Balestrieri, and A. Pardo, “A staged embedding of attribute grammars in haskell,” in *Proceedings of the 30th Symposium on Implementation and Application of Functional Languages*, Lowell MA USA: ACM, Sep. 5, 2018, pp. 95–106, ISBN: 978-1-4503-7143-8. DOI: 10.1145/3310232.3310235. [Online]. Available: https://dl.acm.org/doi/10.1145/3310232.3310235 (visited on 10/22/2022).

[15] M. Grebe, D. Young, and A. Gill, “Rewriting a shallow DSL using a GHC compiler extension,” in *Proceedings of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, Vancouver BC Canada: ACM, Oct. 23, 2017, pp. 246–258, ISBN: 978-1-4503-5524-7. DOI: 10.1145/3136040.3136048. [Online]. Available: https://dl.acm.org/doi/10.1145/3136040.3136048 (visited on 10/22/2022).

[16] N. Valliappan, R. Krook, A. Russo, and K. Claessen, “Towards secure IoT programming in haskell,” in *Proceedings of the 13th ACM SIGPLAN International Symposium on Haskell*, Virtual Event USA: ACM, Aug. 27, 2020, pp. 136–150, ISBN: 978-1-4503-8050-8. DOI: 10.1145/3406088.3409027. [Online]. Available: https://dl.acm.org/doi/10.1145/3406088.3409027 (visited on 10/22/2022).

[17] A. Ekblad, “High-performance client-side web applications through haskell EDSLs,” in *Proceedings of the 9th International Symposium on Haskell*,

Nara Japan: ACM, Sep. 8, 2016, pp. 62–73, ISBN: 978-1-4503-4434-0. DOI: 10.1145/2976002.2976015. [Online]. Available: https://dl.acm.org/doi/10.1145/2976002.2976015 (visited on 10/22/2022).

[18] P. Thiemann, "An embedded domain-specific language for type-safe server-side web scripting," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 1–46, Feb. 2005, ISSN: 1533-5399, 1557-6051. DOI: 10.1145/1052934.1052935. [Online]. Available: https://dl.acm.org/doi/10.1145/1052934.1052935 (visited on 10/22/2022).

[19] R. Evans, S. Frohlich, and M. Wang, *CircuitFlow: A domain specific language for dataflow programming (with appendices)*, Nov. 24, 2021. arXiv: 2111.12420[cs]. [Online]. Available: http://arxiv.org/abs/2111.12420 (visited on 10/22/2022).

[20] A. Mizzi, J. Ellul, and G. Pace, "D'artagnan: An embedded DSL framework for distributed embedded systems," in *Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018*, Vienna, Austria: ACM Press, 2018, pp. 1–9, ISBN: 978-1-4503-6355-6. DOI: 10.1145/3183895.3183899. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3183895.3183899 (visited on 10/22/2022).