Towards Naturalistic EDSLs using Algebraic Effects

Jonathan Immanuel Brachthäuser
University of Tübingen, Germany

Abstract
Domain specific programming languages bridge the linguistic and conceptual gap between domain languages and implementation languages. One aspect of bridging the gap is to express the domain specific concepts in a language more natural for the domain experts. In recent years, in linguistics, concepts from computer science such as effect operations (e.g., shift/reset and continuations in general) have successfully been used to provide compositional models for natural language semantics. We propose to pick up the old theme of naturalistic DSLs and reevaluate it in the scope of algebraic effects. Building on the insights of linguists, we demonstrate how linguistic features such as anaphora, quantification and implicature can directly be implemented in the Dotty programming language using a library for algebraic effects. As opposed to ad hoc implementation techniques for naturalistic DSLs, systematically using algebraic effects and effectful syntax leads to programs that exactly communicate the usage of linguistic features in their types, offers improved error reporting and better IDE support. We believe that effectful syntax opens up a new interesting perspective on the design and implementation of naturalistic DSLs.

ACM Reference format:
DOI:

1 Introduction
Bridging the conceptual gap between domain and implementation language by designing domain specific languages (DSLs), which are closer to the natural language, is an idea as old as it is controversial. Natural languages have the reputation of being lexically and syntactically ambiguous, having complicated and context dependent binding structures and often a non-trivial semantics, which rarely is compositional. In short, attributes programmers don’t like and programming language designers are at best fascinated by. In consequence, many domain specific languages are still far away from being close to natural language. This is in particular the case for DSLs, which are embedded into a general purpose language. With embedded DSLs the host language additionally imposes its own syntactical restrictions and typing discipline on the DSL designer. While each of those limitations is addressed in its own line of work (e.g., syntax extensions as libraries, domain specific type system extensions) our focus here is on linguistic constructs, which are usually neglected since they are inherently non-context-free.

In about the last decade, many developments in modelling the semantics of natural languages have been inspired by computer science and the theory of abstract machines and control operators in particular. Delimited continuations (using for instance Danvy and Filinski’s shift and reset (1990)) have successfully been used to model quantification (“John loves every women”), focus (“John loves Mary”) and polymorphic coordination (“John and Mary left”) (Barker and Shan 2004; Shan 2004, 2005).

Maršík and Amblard (2016) very recently used algebraic effects with handlers to give a compositional semantics to deixis (“John loves me”), quantification with scope islands and implicature (“John, my best friend, loves me”). Algebraic effects (Plotkin and Power 2003) with handlers (Bauer and Pretnar 2015; Plotkin and Pretnar 2009), described in engineering terms, separate the declaration of effectful operations and their usage from the implementation in effect handlers. Typically, to implement an effect operation the effect handlers obtain access to the delimited continuation, that is, the remaining program after the effect usage up to the handler. Compared to control operators, algebraic effects with handlers closely correspond to multiprompt delimited continuations (Kiselyov and Sivaramakrishnan 2016). Roughly, every handler introduces a prompt marker and an effect operation constitutes a shift up to the corresponding handler-prompt. In the context of natural language semantics, shifting to different handlers is essential to enable disambiguation of quantifiers (Barker 2002) and combining different scoping constructs in a single sentence (Maršík and Amblard 2016).

While using insights from linguistics for programming language design is an old idea (Lopes et al. 2003), in this talk proposal, we propose to reconsider the current approach of design and implementation of embedded DSLs by taking recent developments of computer linguistics and algebraic effects into account. We conjecture that algebraic effects are a solid foundation to implement novel syntactic features for embedded naturalistic DSLs, which we refer to as effectful syntax.

2 Effectful Syntax – Examples from Natural Language
To provide some context and support our conjecture, this section shows examples from natural language semantics (Maršík and Amblard 2016), implemented in Dotty (the Scala of the future1) using a library for algebraic effects (Scala-Effekt2). This should serve only as a first example domain. The idea of effectful syntax goes beyond the domain of natural languages and we plan to collaboratively explore other domains with the workshop participants3. Our experience in implementing these examples makes us believe that effectful syntax is (i) modular – linguistic effects and handlers can be encapsulated in modules, separated from the remaining syntax (ii) learnable – separating linguistic effects from domain concepts allows the DSL user to learn both separately, also strong (effect) typing enables better error messages for wrong usage of linguistic constructs (iii) maintainable – user programs concisely express

1http://dotty.epfl.ch
2http://b-studios.de/scala-effekt
3The full code and an interactive programming environment is available online at: https://scastie.scala-lang.org/tefQ4o6CHQmnKfsCrKwTfNWg.
the use of linguistic features in their (effect) types, opening up opportunity for refactorings and IDE support.

Figure 1a defines the syntax of our EDSL for sentences. The type constructor $C$ (pronounced "control") represents effectful computation, and is a monad provided by the Effekt library with the corresponding monadic methods and properties. To also allow effectful semantics of the EDSL, all return types are wrapped in $C$. In the definition of linguistic effects (Figure 1b) and in user programs, we assume $S$ and $NP$ to be the type used for the semantics domain of sentences and nominal phrases, respectively.4

**The Speaker Effect.** We begin with a simple sentence that uses the speaker effect to refer to the contextual speaker of the sentence:

```scala
trait Sentences[NP, S] {
  def person(name: String): C[NP]
  def man(person: NP): C[S]
  def woman(person: NP): C[S]
  def loves[src: NP, trg: NP]: C[S]
  def bestFriendOf [friend: NP, person: NP]: C[S]
  def said(person: NP, sentence: S): C[S]
  def forall[f: NP ⇒ C[S]]: C[S]
  def and[first: S, second: S]: C[S]
}
```

(a) Syntax of the Sentences EDSL as a shallow embedding.

Running the example sentence $s_1$ with speaker Pete, shows how the annotated apposition is lifted up to the accommodate handler and introduces a conjunction.

```scala
val s4 = C[S] = john bestFriendOf pete, loves(mary, John))
```

The sentence $s_4$ also shows that multiple linguistic effects can naturally be combined. The type of statements expresses which effects are not yet handled. The idea of effectful syntax is of course completely independent of Scala and the sentence DSL could similarly be implemented in other languages that support algebraic effects and handlers. However, since we embedded the DSL into Scala, we could customize the compile time errors, when effects are not handled. This customization might be an interesting feature addition for languages with native support for algebraic effects like Koka (Leijen 2014).

3 Conclusions

In the talk, we will suggest to consider algebraic effects for the design and implementation of more naturalistic EDSLs. We hope that our proposal triggers a vivid discussion about other application domains and sparks new research opportunity to develop novel abstraction mechanisms for effectful syntax, assisting the design and maintenance of naturalistic DSLs.

References