Domain Specific Languages
5: Are DSLs domain specific?

Greg Michaelson
School of Mathematical & Computer Sciences
Heriot Watt University
DSLs and expressiveness

• DSL’s alleged to have greater expressiveness than general purpose languages

• claimed DSLs can express:
  – same things as other languages, but more succinctly
  – things other languages can’t

• notion of expressiveness is comparative
What is expressivity?

Rob Stewart
@robstewartUK

Please RT! Expressivity of a programming language: the ability write a small/succinct program, or the ability to write a program?

55% A small/succinct program

45% A program

165 votes • Final results

RETWEETS
17

9:31 AM - 5 Jan 2017

6 17
Formalising expressiveness


• suppose language X has constructors which are not in language Y

• language X is a *conservative extension* of language Y if:
  – instances of X can be translated into instances of Y without changing semantics of Y
Formalising expressiveness

• e.g. GpH adds \texttt{par} & \texttt{seq} constructs to Haskell
  – conservative extension
  – simple elimination of \texttt{seq} & \texttt{par} preserves meanings of programs

• \textit{weak expressibility}
Formalising expressiveness

• if constructors in X cannot be eliminated in translation to Y then:
  – X has semantic properties Y lacks
  – X extends Y
• e.g. pure functional Scheme based on LISP
• cannot express state changing assignment `set!`
• LISP extends Scheme
Embedded DSL (EDSL)

- DSL embedded in extant host language

1. write functions in host language
   - call directly
   - can use arbitrary host language constructs
   - not a distinct language
   - shallow embedding
Embedded DSL

2. add abstract syntax
   • interpret ASTs
   • call interpreter components from arbitrary host language constructs
   • not a distinct language unless only use ASTs to program
   • deep embedding
   • conservative extension
Embedded DSL

3. design concrete syntax
• add concrete syntax -> AST compiler
• call interpreter component with parsed strings
• can embed interpreter component calls in arbitrary host language constructs
• not a distinct language unless only use concrete syntax strings
• deep embedding
• conservative extension
Embedded DSL

4. extend host language syntax
   • conservative extension

5. extend host language semantics
   • extension
Turing completeness

• Hilbert’s program
  – can number theoretic predicate calculus establish its own completeness & consistency?
    • no – Gödel - 1932
  – can theorem-hood be established mechanically?
    • no - Turing & Church - 1936
Turing completeness

• need a system for formalising “mechanical”
• models of computation
  – Church
    • algorithm/effectively calculable
    • λ calculus
  – Turing
    • computable
    • Turing machines
• Turing proved these equivalent 1936/37
Turing completeness

• Church-Turing thesis
  – all models of computation are equivalent
  – demonstrate by constructing/proving translations both ways between known & new systems
• system that satisfies C-T Thesis is Turing complete (TC)
• e.g. von Neumann machines $\rightarrow$ digital computers
• e.g. programming languages
EDSL & host language

- EDSL inherits semantics of host language
- If host language is not TC then EDSL may be more expressive
  - i.e. EDSL == host + external library in 3rd language
  - Can’t be less or differently expressive
- If host language is TC then so is EDSL
  - Not domain specific?
EDSL & host language

• if host language exposed to programmer...
• ...then programmer can deploy arbitrary host language constructs
• not domain specific?
Implemented DSL

• expose parser/interpreter only as *stand alone* language processor
• can only use domain specific syntax
• have *implemented* DSL in host language
DSL bloat

• tend to want familiar general purpose programming language abstractions as well, so add:
  – arithmetic & logic
  – sequence/selection/iteration
  – sub programs
  – data structures
• as DSL grows, tends to become:
  – more and more like favourite language
  – less and less DS
Language & program

- language = syntax + semantics
- semantics: program * state -> state
- semantics transforms initial state to final state guided by structure of program instance
Language & program

• program: input * state -> output * state
• treat outputs as part of final state
• program: input * state -> state
• program changes initial state to final state depending on input structure
• define input structure with syntax
• program is semantics of inputs
• does every program define a DSL?
What is domain specificity?

• Light Bulb Language

program -> switch
switch -> SWITCH | ε
m [SWITCH] ON = OFF
m [SWITCH] OFF = ON
m [ε] ON = ON
m [ε] OFF = OFF
What is domain specificity?

• Linear Light Bulbs Language

\begin{align*}
  program & \rightarrow \text{row} \mid \text{row program} \\
  row & \rightarrow \text{switch} \mid \text{switch row} \\
  s_i & \in \text{row} \\
  b_i & \in \{\text{ON,OFF}\} \\
  m' \ [s_1,\ldots,s_N] \ {b_1,\ldots,b_N} & = \{m \ s_1 \ b_1,\ldots,m \ s_N \ b_N\}
\end{align*}
What is domain specificity?

• Grid Light Bulbs Language

program -> grid
grid -> row | row grid

\[ r_i \in \text{row} \]
\[ \text{br}_i \in b^* \]

\[ m'' \{ \text{br}_1, \ldots, \text{br}_N \} = \{ m' \ p \ \text{br}_1, \ldots, m' \ p \ \text{br}_N \} \]
What is domain specificity?

• is GLBL domain specific...?
  – light bulbs?
  – B/W images?
  – anything representable as a Boolean?
• not very DS...
What is a domain?

• computer pioneers thought machine code was for configuring hardware

• 60’s language designers thought languages were purpose specific:
  – FORTRAN - sums
  – COBOL - accounts
  – ALGOL - algorithms
  – LISP - symbols
  – BCPL - systems
What is a domain?

1962
What is a domain?
What is a domain?
What is a domain?
What is a domain?
What is a domain?
What is a domain?
What is domain specificity?

• all TC languages capture common notion of computation
• TC languages with different semantics are mutually extending
• do all TC languages have the same expressiveness?
What is domain specificity?

• designers of new languages think they’ve enabled something other languages can’t do
• new TC language ≡
  old TC language + syntax + library
• is every language really an embedded DSL with a TC host?
What is domain specificity?

• Felleisen suggests comparisons in a common *language universe*

• some TC languages can express some algorithms more succinctly than other TC languages

• which language universe?

• do language universes have language biases?
DSL is about pragmatics

• Felleisen:
  – “...what advantages there are to programming in the more expressive language when equivalent programs in the simpler language already exist.”
  – “...programs in less expressive languages exhibit repeated occurrences of programming patterns and this pattern oriented style is detrimental to the programming process.”
  – “Conciseness conjecture. Programs in more expressive languages that use the additional facilities in a sensible manner contain fewer programming patterns than equivalent programs in less expressive languages.”
DSL is about pragmatics

• DSL abstractions & constructs make it easier to express particular things
• what may be complex in an arbitrary TC language may become simpler in a DSL
• domain may frame choice of DSL abstractions & constructions
• abstractions and constructions from one domain may be appropriate for other domains