The Design and Use of Domain Specific Languages

Simon Goldrei
simon@cs.usyd.edu.au
Standalone - Compiler/Interpreter

• Advantages
  – Syntax close to notation used by domain experts
  – Useful error reporting possible
  – Domain-specific verification, optimisation, algorithmic reductions
  – Domain-specific static type checking

• Disadvantages
  – Requires development of a language processor
  – Risk of inconsistent design
  – Difficult to extend with DSL evolution
  – Perhaps not as expressive or elegant

• Focused on domain Syntax
LaTeX

• A DSL for Authors
  – LaTeX encourages authors *not* to worry too much about the appearance of their documents, but to concentrate on getting the right content

\layout Title
The Design, Implementation and Use of Domain Specific Languages
\layout Author
Simon E. C. Goldrei
newline
School of Information Technologies, University of Sydney
\begin_inset Foot collapsed false
On exchange at the Baskin School of Engineering, University of California Santa Cruz
\end_inset

SECG 2004 - CMPS203
Embedded (EDSL/DSEL)

• Advantages
  – Less development effort, no language processor to implement
  – A more feature rich language
  – Re-use of existing host language tools
  – Users may already be familiar with host language

• Disadvantages
  – Syntax not in terms of domain
  – Operator overloading can become confusing
  – Poor, misleading error reporting
  – Poor performance common particularly in functional languages, poor optimisations

• Focused on domain Semantics
A Musical Interlude…

- **Haskore (Has Score) circa 1996 - 2000**
  - DSL for Musical structures eg:
    - Notes, Rests, Transposition, Tempo
    - Music, Players, Performance

  ```haskell
  f :: Int -> (Chord -> Chord)
  ds = [ (III,1), (VI,2), (VI,3), (II,4), (II,5), (Vb,6),
         (V,7), (Ib,8), (I,9), (IV,10) ]
  ds' = map (\(x,y) -> (y,x)) ds
  dd h = lookup h ds
  dd' x = just (lookup x ds')
  ```

- **A chord may be followed by a chord a third below, a second above, a fourth above or by itself**

  ```haskell
  f 2 s = dd' ( maximum [1, ((maximum (dd s)) - 6)] )
  f 3 s = dd' ( maximum [1, ((maximum (dd s)) - 4)] )
  f 4 s = dd' ( minimum [10,((maximum (dd s)) + 2)] )
  f 5 s = s
  ```
Motivations for DSL Development

• Improve the economics of repeated software development tasks
• Cater to the domain engineer or allow end-user specification:
  – Software Re-use
  – Maintainability
  – Repetitive tasks
• Provide domain specific program verification, optimisation, algorithmic transformations or reductions
In Conclusion

- Compiler/Interpreter
  - Domain notation strict
  - Domain optimisations
  - Domain Error reporting

- Embedded DSL
  - All other cases
  - Limited resources
  - Familiarly with language
    - PERL for biologists