4 Domain Analysis
Context of Domain Analysis

- Select and define domain of focus
- Collect relevant domain information
- Integrate domain information into a model/language
Domain Model

- Defines **scope** of the domain
- Characterizes **content** by examples and counter examples
- Defines **domain vocabulary** (domain lexicon)
- Describes **domain concepts** using a modeling formalism (E.g.: object model, E/R model, data-flow diagram, grammar + mappings)

In this class:

- **Haskell DSEL**

- **DSL Report Card**
Use Cases

**Use case:** Set of scenarios with a common user goal

Use cases are a helpful tool to capture requirements
Use-case diagrams are not really needed
Domain Decomposition

Domain Analysis

Domain Decomposition

Domain Modeling

Semantics-Driven DSL Design

Syntactic Design

Domain Realm

Metalanguage Realm

Language Schema

Language

Relationship

Domain
Domain Decomposition

Domain Model

- Domain
- DSL
- Data Type
- Type Constructor
- Relationship

Domain Realm

Semantics-Driven DSL Design

Syntactic Design

Haskell
Example: Labeled Pictures

First Attempt:
Direct representation

data Pic a = Line a Point Point
           | Circle a Point Int
           | Pic a :+: Pic a

Generic label type
Label redundantly replicated over all constructors
design too monolithic
Domains into Languages

Domain Decomposition

\[ D = R(D1, D2) \]

Domain Modeling

\[ D = R(D1, D2) \]

Language

\[ \text{data } L1 = \ldots \]

Language Schema

\[ \text{data } S \ a \ b = \ldots \]

Language

\[ \text{data } L1 = \ldots \]

Language

\[ \text{type } L = S \ L1 \ L2 \]
Separating Labels

Domain Decomposition

Label

Attach

Picture

type Label = String

data Attach a b = Att a b

data Pic = Line Point Point
  | Circle Point Int
  | Pic :+: Pic

type LPic a = Attach a Pic

What's wrong here?
Refined Labeling

Domain Decomposition

Label

Attach

Domain Modeling

Figure

Overlay

Picture

type Label = String

data Attach a b = Att a b

data Fig = Line Point Point | Circle Point Int

type LFig a = Attach a Fig

data Pic a = Base LFig | Pic a :+: Pic a

data Pic = Int :+: Pic

Design Guidelines for Domain Modeling

- **Redundancy** (in types) indicates need for further decomposition
- **Arbitrary choices** (for types) indicate the need for parameterization
- **Minimalist Design**: Include only essential features

**Type Constructor** = Language Schema

**Parameterization** = Generalization