4 Domain Analysis
Context of Domain Analysis

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

In this class:

- Haskell DSEL
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:
- DSL Report Card
- Haskell DSEL
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

Domain Realm

D1

R

D2

Domain Decomposition

Semantics-Driven DSL Design

DSL

Syntactic Design

S

L1

L2

Metalanguage Realm

Domain

Language

Language Schema

Relationship

Domain-Driven DSL Design

Domain Modeling
Domain Decomposition

Domain Analysis
Example: Labeled Pictures

First Attempt: Direct representation

```haskell
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

\[ D = R(D_1, D_2) \]

**Domain Decomposition**

**Domain Modeling**

**Language Schema**

**Language**

**Language**

\[ \text{data } L_1 = \ldots \]

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

\[ \text{data } L_1 = \ldots \]

\[ \text{type } L = S \ L_1 \ L_2 \]
Separating Labels

Domain Decomposition

Label

Attach

Picture

Domain Modeling

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?
**Domain Analysis**

Refined Labeling

**Domain Decomposition**

- **Label**
  - **Attach**
    - **Figure**
      - **Overlay**
        - **Picture**

**Domain Modeling**

- **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 a) | Pic a :+: Pic a**
- **data Pic = Base LFig | Pic :+: 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

Parameterization = Generalization

Type Constructor = Language Schema