CSSE 490 Model-Based Software Engineering: Transformational Programming

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

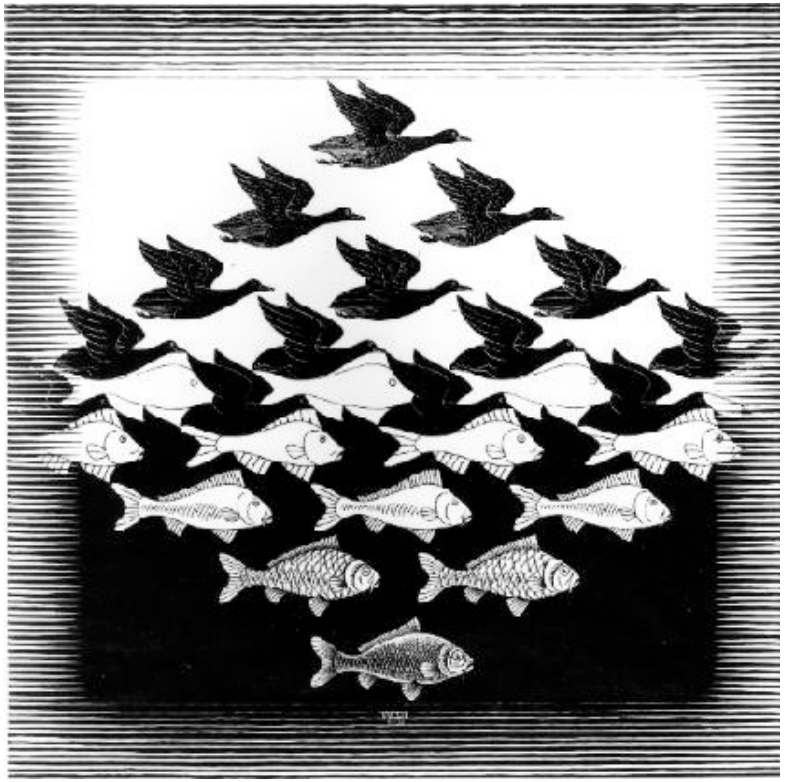
Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Learning Outcomes: Transformations

Define transformation rules for abstraction and refinement.

- Outline transformation systems
- Introduce transformational programming
- Discuss paper (if time)



For all the SQL calls in a 2M line program done in MySQL, how would you rewrite them in Oracle SQL? (not the details, just the strategy)

- Think for a minute…
- Turn to a neighbor and discuss it for a minute

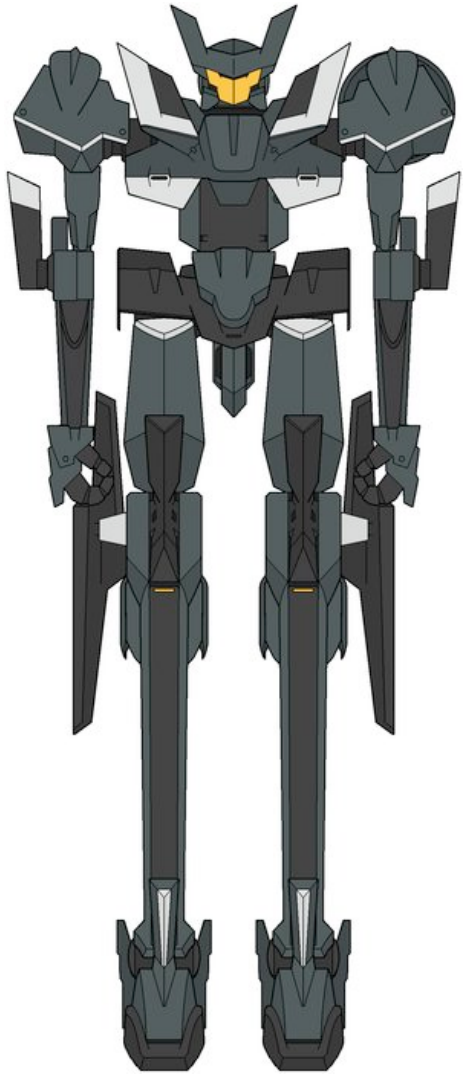
Rules need verifiers…

Transformational System Applications

- General support for program modification
- Program synthesis from a formal specification
- Automatic program generation
- Adaptation to different environments
- Verification of program correctness

Transformation Systems

- Correct programs can be built if the task is split into sufficiently small and formally justified steps
- Many of those steps are automatable
- If the automatable steps are performed by a machine, the programmer is free to focus on creative aspects of the job!



Transformation System Issues

- Specification vs. programming languages
- Level of automation – full, semi, user-driven
- Transformation mechanism approaches

  Catalog approach:
  Production rules, knowledge-based systems

  Generative set approach:
  Elementary transformations used in constructing new rules

Types of Transformational Systems

- Restructuring/Optimization
  - Same input and output language
- Conversion/Synthesis
  - Different input and output language

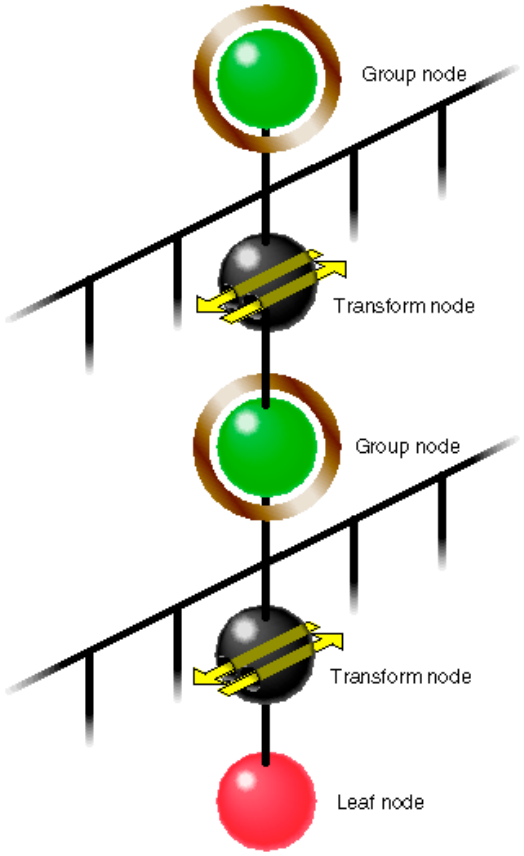
Transformational Programming

- Programming by successive application of transformation rules
- **Transformation** — a relation between two programs
- **Transformation rule** — mapping from one program to another that constitutes a correct transformation (e.g., equivalency)

- Guarantees that the final version of the program satisfies the initial formal specification

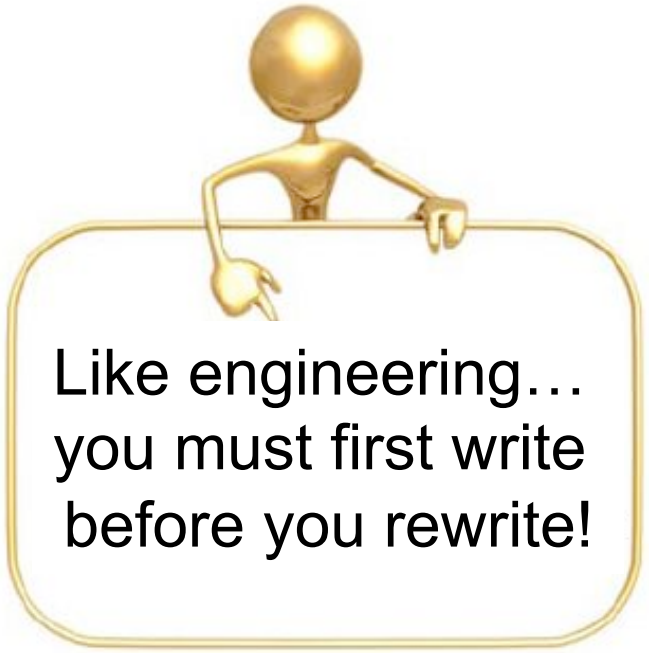
Specifying Basic Transformation Steps

- Rewrite rules
  - Substitution
  - Pattern Matching
  - Rule application
- Examples
  - Propositional formulae
  - Lambda calculus



Rewrite Rules

- Rule: `L : l → r`
  - Label/name `L`
  - Left-hand side pattern `l`
  - Right-hand side pattern `r`

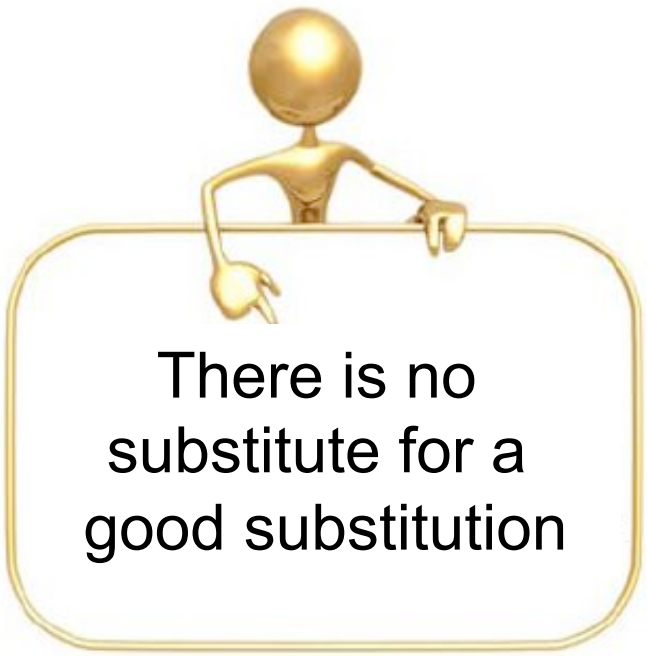


- Pattern: term with variables
  - `t := x | C(t1, …, tn) | C | int | string`
- Examples
  - `A : Plus(Zero, x) → x`

Substitution

- A *substitution* is a mapping from variables to terms

- Notation: `[t1/x1, … , tn/xn]` is a finite substitution mapping `xi` to `ti` and all other variables to themselves

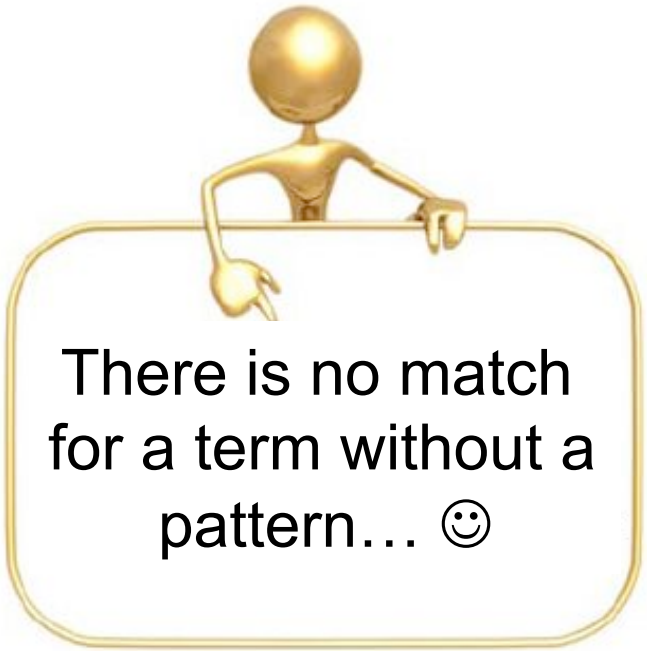


- Application of a substitution `s` to a pattern
  - `subst(s, x) = s(x)`
  - `subst(s, str) = str`
  - `subst(s, num) = num`
  - `subst(s, C(t1, … , tn)) = C(subst(s,t1), … , subst(s,tn))`

Term Pattern Matching

- A term `t` matches with a pattern `p` if there is a substitution `s` such that

```
subst(s,p) = t
```



Application of Rule


```
Rule L : l → r
Term t
Application of L to t
     = subst(s, r),
          if subst(s, l) = t
     = fail, otherwise
```

```
t1 –L→ t2 : L t1 rewrites to t2 under rule L
```

```
t1 → t2 : t1 rewrites to t2 under some rule
```

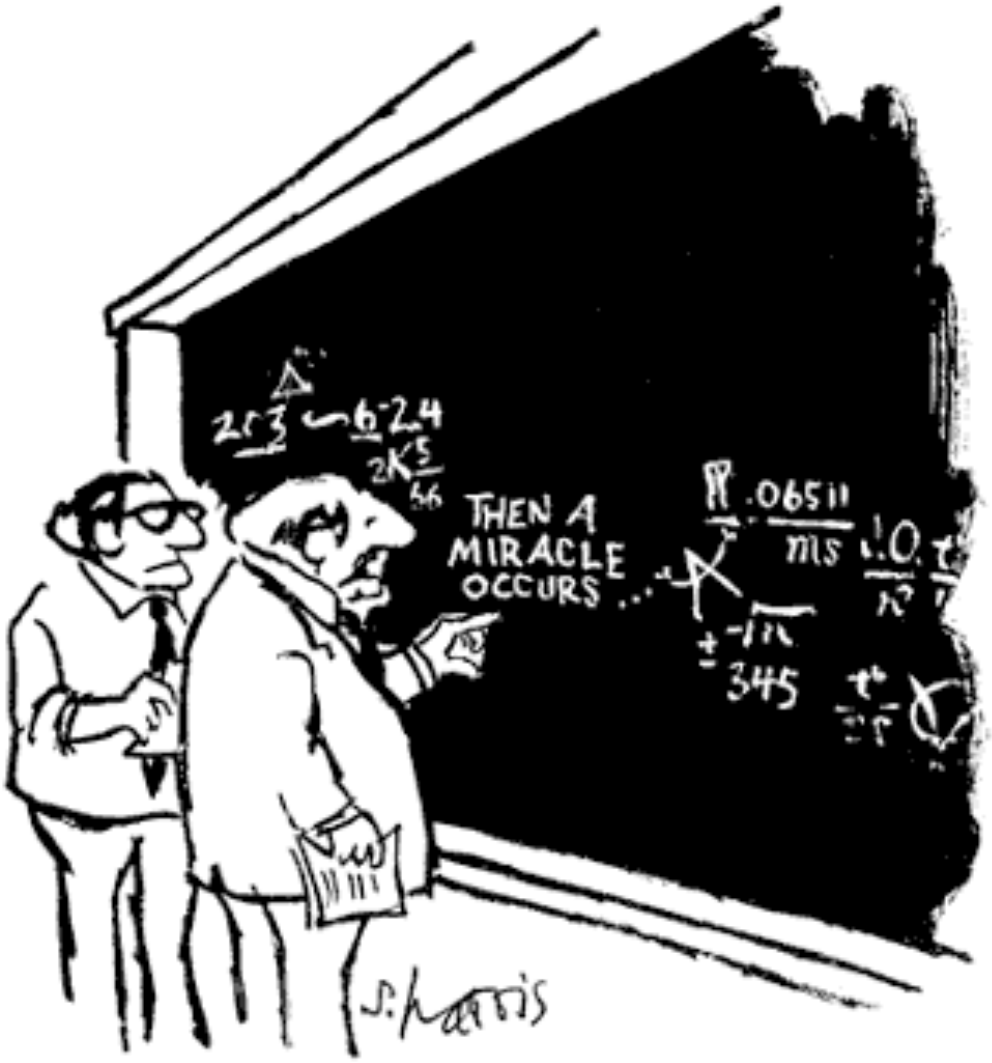
Prop: Algebraic Simplification

```
Module prop-laws imports prop
Rules
   // Associativity relation

   AA : And(And(x, y), z) → And(x, And(y, z))
   AO : Or(Or(x, y), z) → Or(x, Or(y, z))
   AI : Imp(Imp(x, y), z) → Imp(x, Imp(y, z))
   AE : Eq(Eq(x, y), z) → Eq(x, Eq(y, z))

   // Commutative relation

   CA : And(x, y) → And(y, x)
   CO : Or(x, y) → Or(y, x)
   CE : Eq(x, y) → Eq(y, x)
```

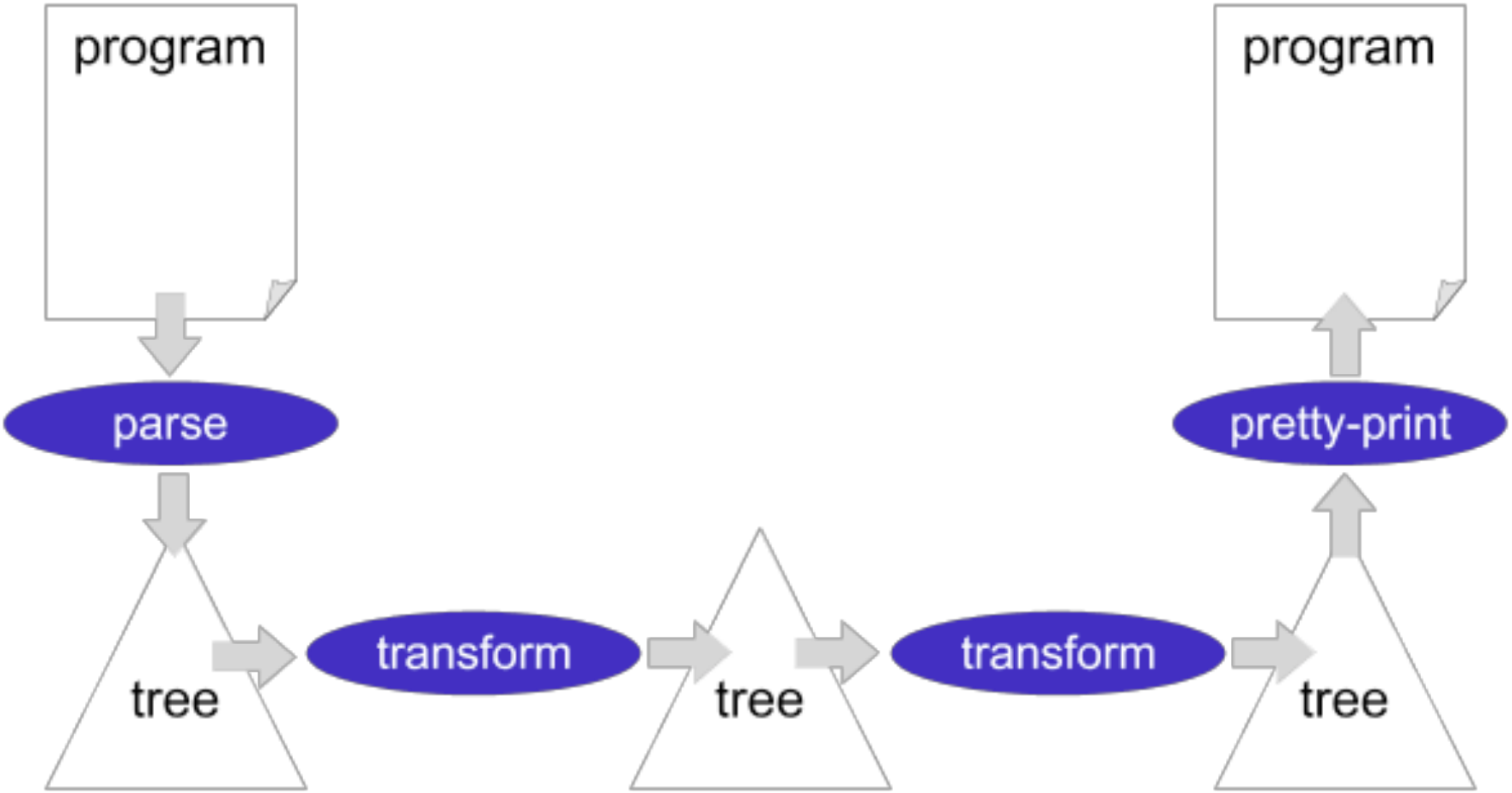


"I think you should be more explicit here in step two."

Program Transformation Paradigms

- Interactive Program Transformation
- Syntactic Abstractions in Intentional Programming
- Simple Tree Parsing
- Tree Parsing with Dynamic Programming
- Term Rewriting
- Term Rewriting with Strategy Annotations
- Functional Rewriting
- Rewriting with Traversal Functions
- Controlling Rewriting by Reflection
- Sequences of Canonical Forms
- Non-deterministic Sequential Strategies
- Generic Traversal Strategies

Simple Transformational System



Example Rewrite Rules

```
InlineF :
 |[ let f(xs) = e in e'[f(es)] ]| ->
 |[ let f(xs) = e in e'[e[es/xs]] ]|

InlineV :
 |[ let x = e in e'[x] ]| -> |[ let x = e in e'[e] ]|

Dead :
 |[ let x = e in e' ]| -> |[ e' ]| where <not(in)> (x,e')


Extract(f,xs) :
 |[ e ]| -> |[ let f(xs) = e in f(xs) ]|


Hoist :
|[ let x = e1 in let f(xs) = e2 in e3 ]| ->
|[ let f(xs) = e2 in let x = e1 in e3 ]|
 where <not(in)> (x, <free-vars> e2)
```

Applications of Software Transformation 1/2

- Compilers
  - Translation (e.g. Java into C#)
  - Desugaring (e.g. Java's foreach into for)
  - Instruction selection
    - Maximal munch vs BURG-style dynamic programming
  - Optimization
    - Data-flow optimization, Vectorization, GHC-style simplification, Deforestation, Domain-specific optimization, Partial evaluation…
  - Type checking
  - Specialization of dynamic typing

Applications of Software Transformation
2/2

- Program generators
  - Pretty-printer and signature generation from syntax definitions
  - Application generation (e.g. data format checkers from specifications)
- Program migration
  - Platform conversion (e.g. MacOS to Linux)
- Program understanding
  - Documentation generation (e.g. JavaDoc)
- Document generation/transformation
  - Web/XML programming (server-side scripts)

So, What does this have to do with MBE?

- Reduces requirements errors as it forces rigor in the requirements analysis
  - Incompleteness and inconsistencies can be discovered and resolved
- Correctness by construction - preserving and guaranteeing essential properties
- Both specification and transformation rely on the rigors of Formal Specification and Transformation

Paper Discussion: Feature-Based Transformation Approach Paper

Feature-based survey of model transformation approaches

- What are the main thrusts of the paper?
- What are the controversial points and your positions?
- What did you get out of reading about feature-based transformation approaches?

Homework and Milestone Reminders

- Continue to familiarize yourself with material on Eclipse Modeling Project

  http://www.eclipse.org/modeling/