Model Verification with TXL and Alloy: An Adaptation of The Experience Report

Tommy Carpenter & David Gage

April 5, 2012

The Rundown

1. Motivation
2. DSL/TXL
3. Alloy Correctness And Demo
4. Complaints & Insights
5. Outro

Problem & Motivation

Here's What We Are Doing

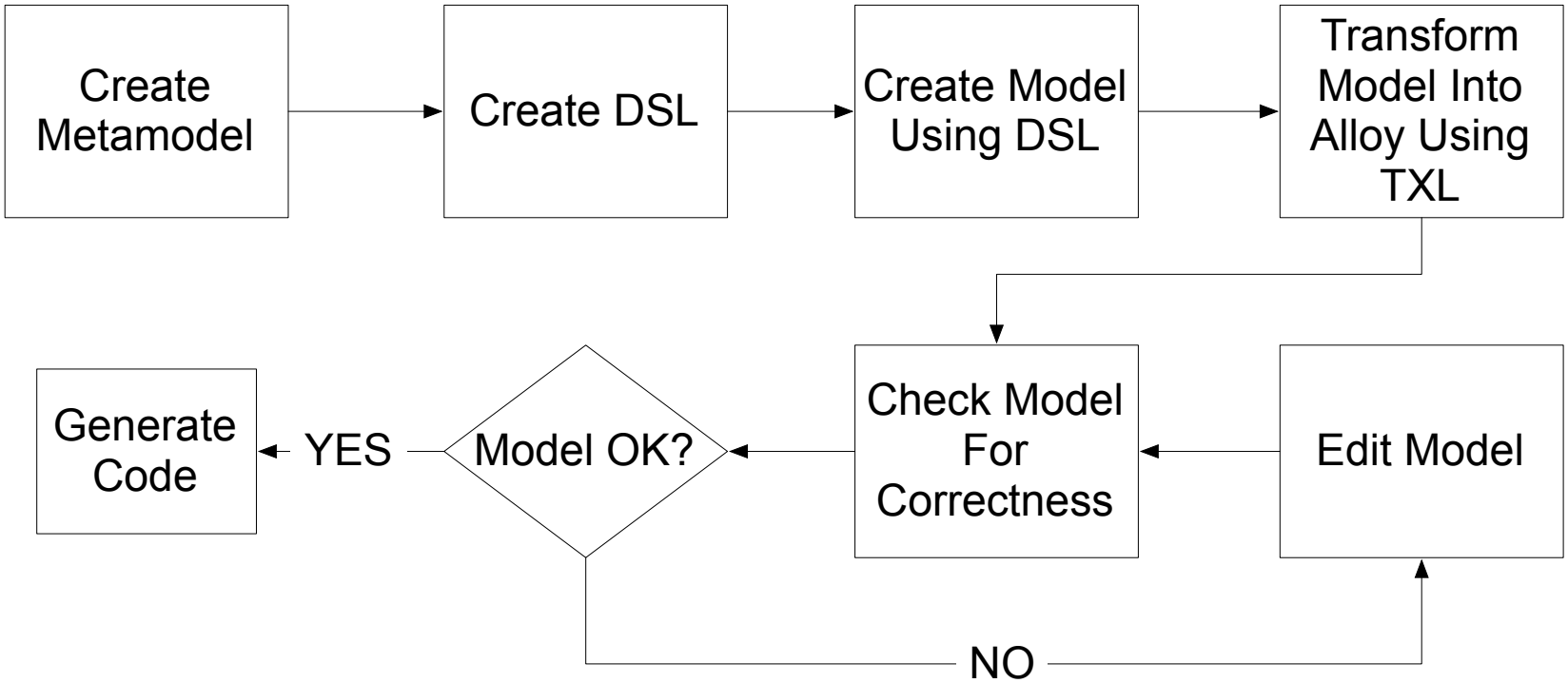
We...

1. Create a *metamodel* describing the BCPU's basic components.
2. Create a DSL to instantiate/link/add functionality to the components defined in the metamodel.
3. Use the DSL to create our *model*.
4. Use TXL to transform our model into Alloy.
5. Use Alloy to verify the model's *correctness*.

Here's Why We Are Doing It

1. Most groups are focusing on code generation.
2. Checking model for correctness prior to code generation = less bugs in code!

Our BCPU MBSE Pipeline



DSL/TXL

DSL Snippet

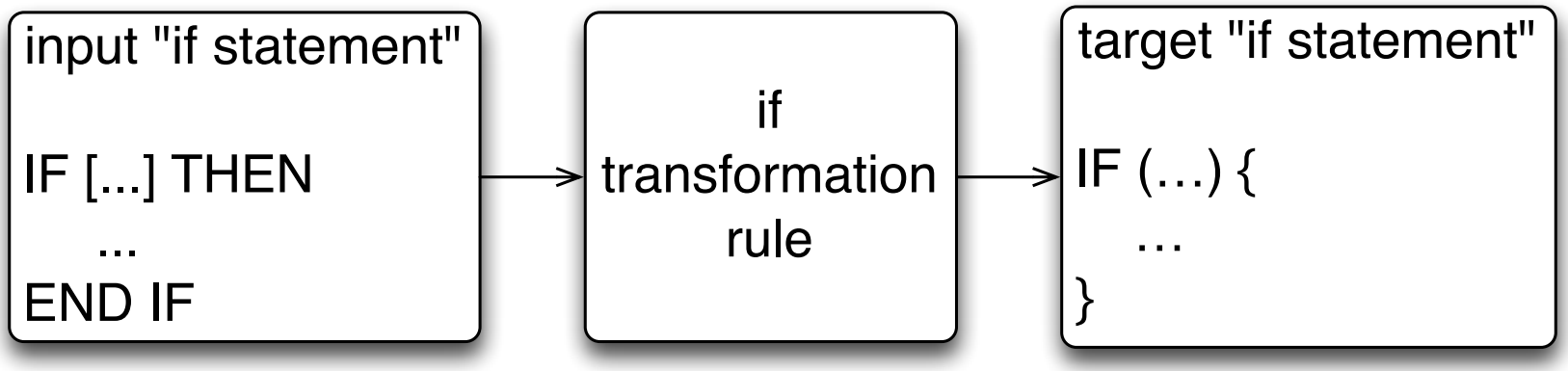
```
new Mode Speed{
        display: speed;
        units: KPH;
        compute: speed = (revolutions*circumference/
1000*360);
}

new Button nextMode{
        action: next();
}
```

TXL Transformation Process

TXL is a source transformation language based on *statement definitions* and *rules*.

1. Describe the structure of the input sources' allowable *statements*.
2. Define the *rules* TXL uses to transform the input source into the target source.



TXL Module Example

```
define Statement
      [numberAssignStatement]
    | [additionStatement]
end define

define numberAssignStatement
    [id] <- [number]
end define

define additionStatement
    [id] <- [number] '+ [number]
end define
```

TXL Rule Example

```
%converts "x <- 5" into "x = 5;"
%V and N are bound to ‘‘x" and "5"
rule transformNumberAssignStatement
    replace [numberAssignStatement]
        V [id] <- N [number]
    by
        V ’= N ’;
end rule
```

Alloy Correctness And Demo

So What Is Correctness Anyway?

With Alloy we can ask:

- Are all necessary components contained in the model?
- Are the BCPU constraints properly implemented in the model?
- Is all functionality defined by the BCPU specs implemented?
- Are there inconsistencies in the original specs of the BCPU?

Alloy Demo (Explained Through Example)

[Farmer River Example]
[Place Working Demo *Or* Funny Cat Pictures Here]

Complaints & Insights

TXL Pros

- Easy to learn the basics; can use for simple cases in mins
- Excellent tutorials online
- Built-in keywords make much of parsing easy; *repeat, number, id...*
- Regex support
- Gives insight into designing languages based on parsing difficulty

TXL Cons

1. More complex translations seem to be hard or impossible
   - example; we have to run two translations to run the alloy code
2. Rules cannot have more than one “replace by” statement:

```
define ifRelationalStatement
     if '( [id] greater than [id] ')
   | if '( [id] less than [id] ')
   | if '( [id] equals [id] ')
end define

rule transformIfRelationalStatement
   replace [ifRelationalStatement]
    if '( X [id] greater than Y [id] ')
   by   if '( X '> Y ')
   replace  if '( X [id] less than Y [id] ')
   by   if '( X '> Y ')
   replace   if '( X [id] equals Y [id] ')
   by   if '( X == Y ')
end rule
```

3. Unison grammars...

Alloy Pros

- Alloy is excellent at telling you what a model can do
- When used correctly, you can ask complex and interesting questions

Alloy Cons

- Steep learning curve; model checking languages are not easy
- Alloy is not great at telling you what a model *is* doing as opposed to what the model *can* do

Outro

- Model checking is useful, but hard
- TXL is a decent tool for source transformation, but is limited. Very good for similar languages, not good for complex transitions.
- Just scratching the surface of Alloy..