Searching for Strategies that Verify MDE Toolchains

Simon Poulding, University of York & DAASE
collaboration with Louis Rose, University of York

Context

Approach

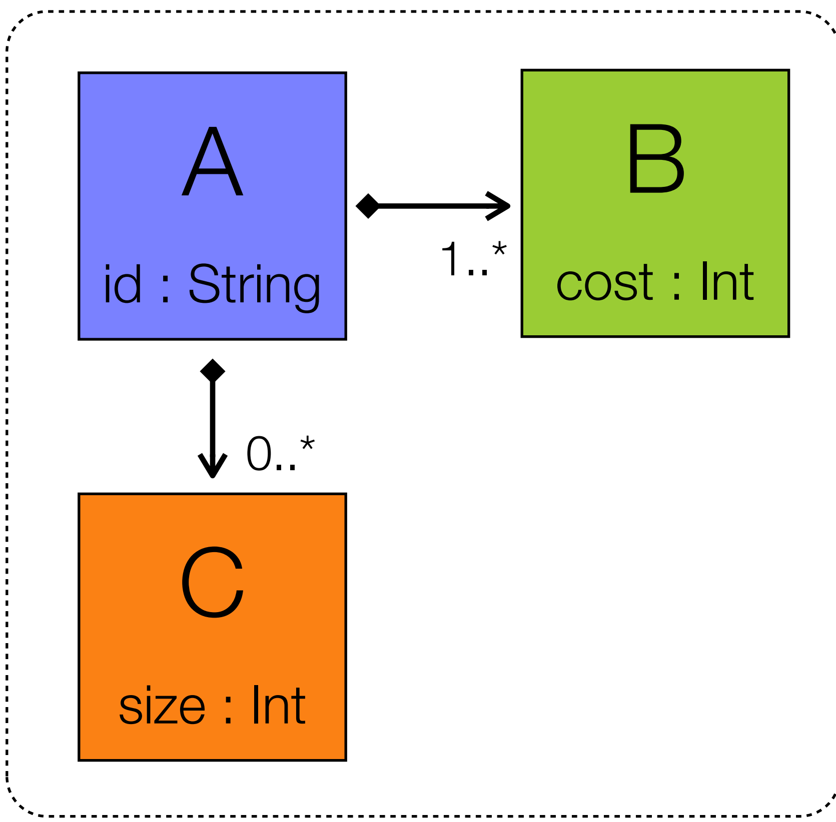
Implementation

Case Study

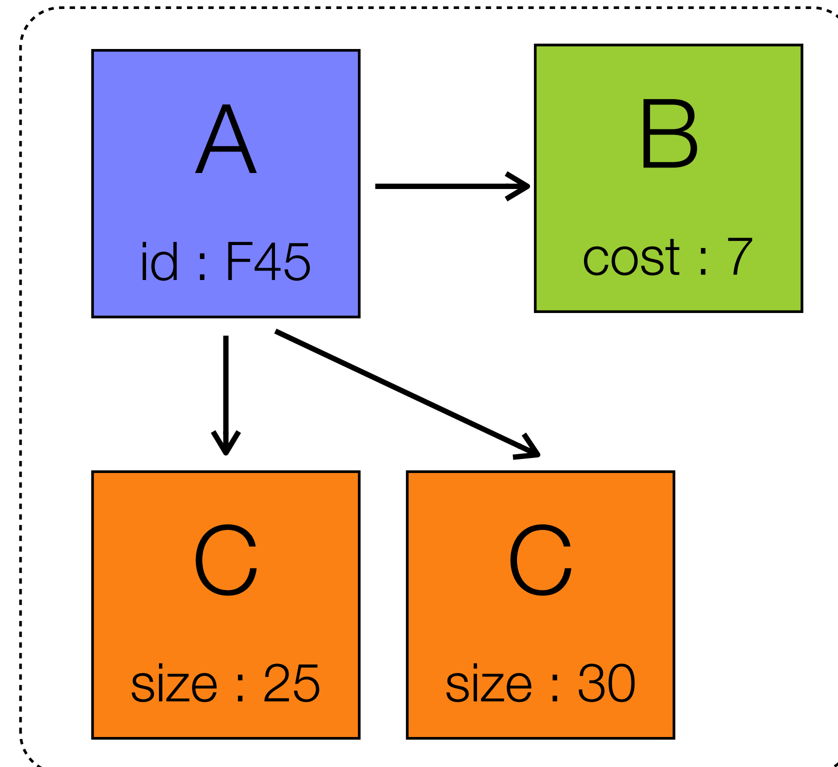
Metamodels

In Model-Driven Engineering (MDE), model instances must comply to a metamodel that specifies attributes and associations

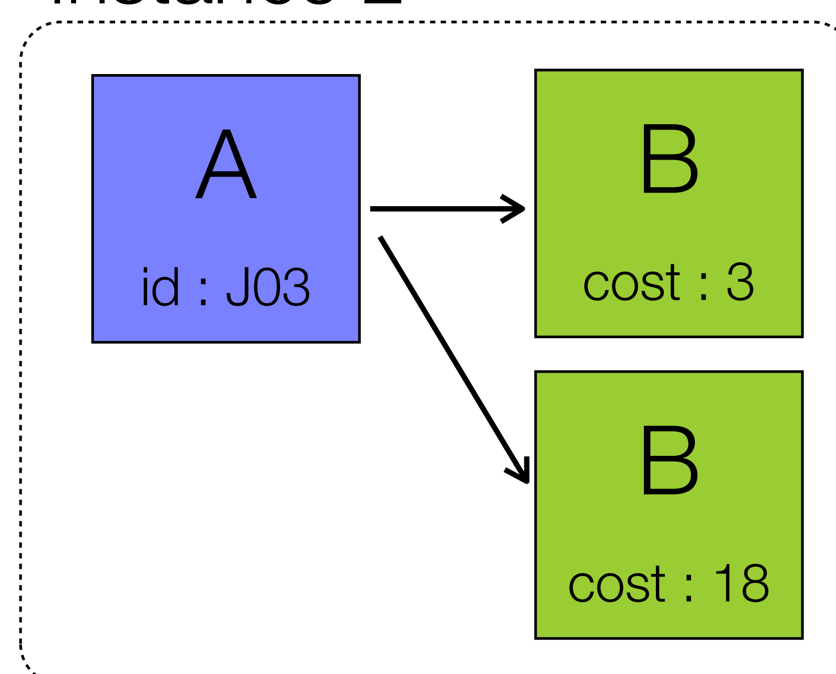
Metamodel



Instance 1

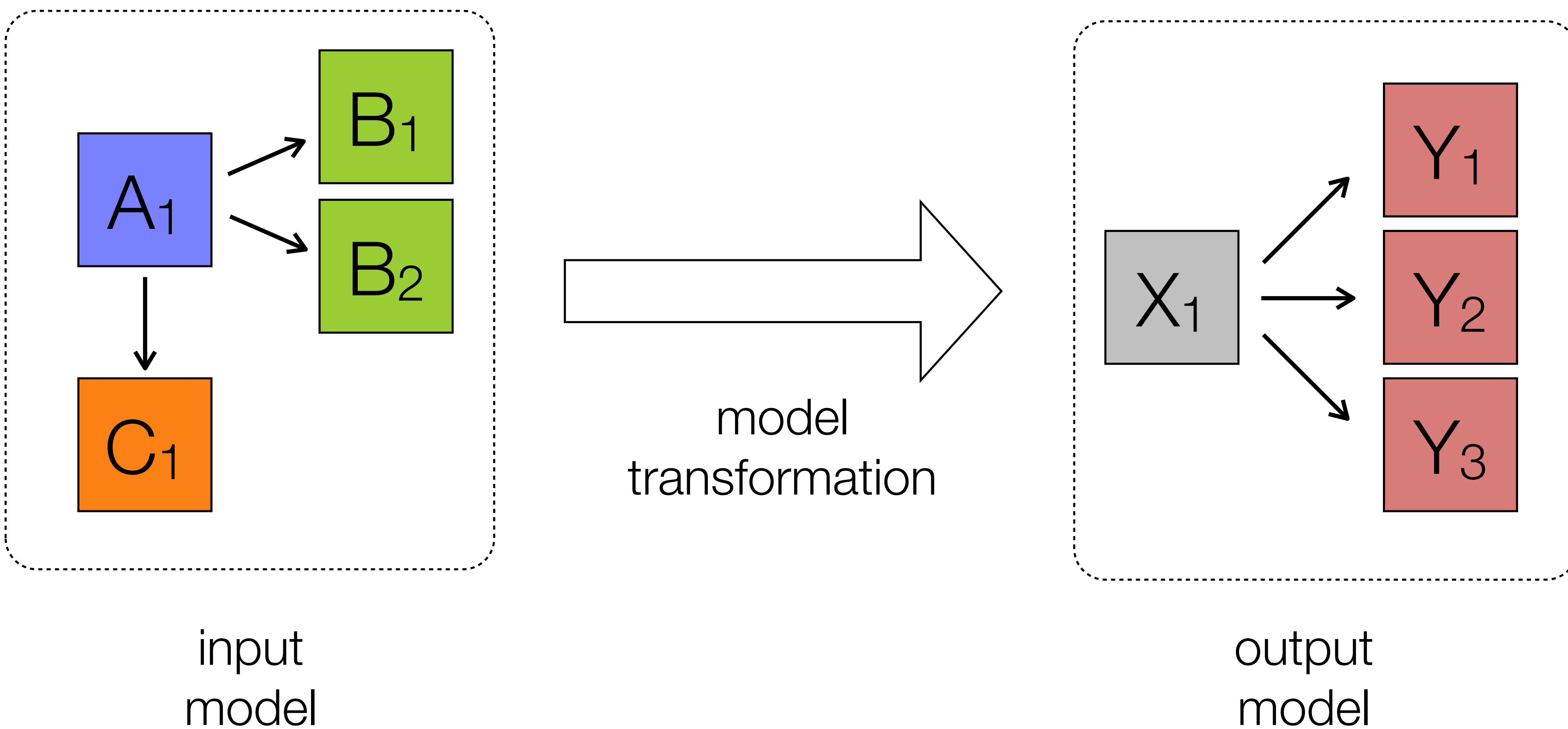


Instance 2



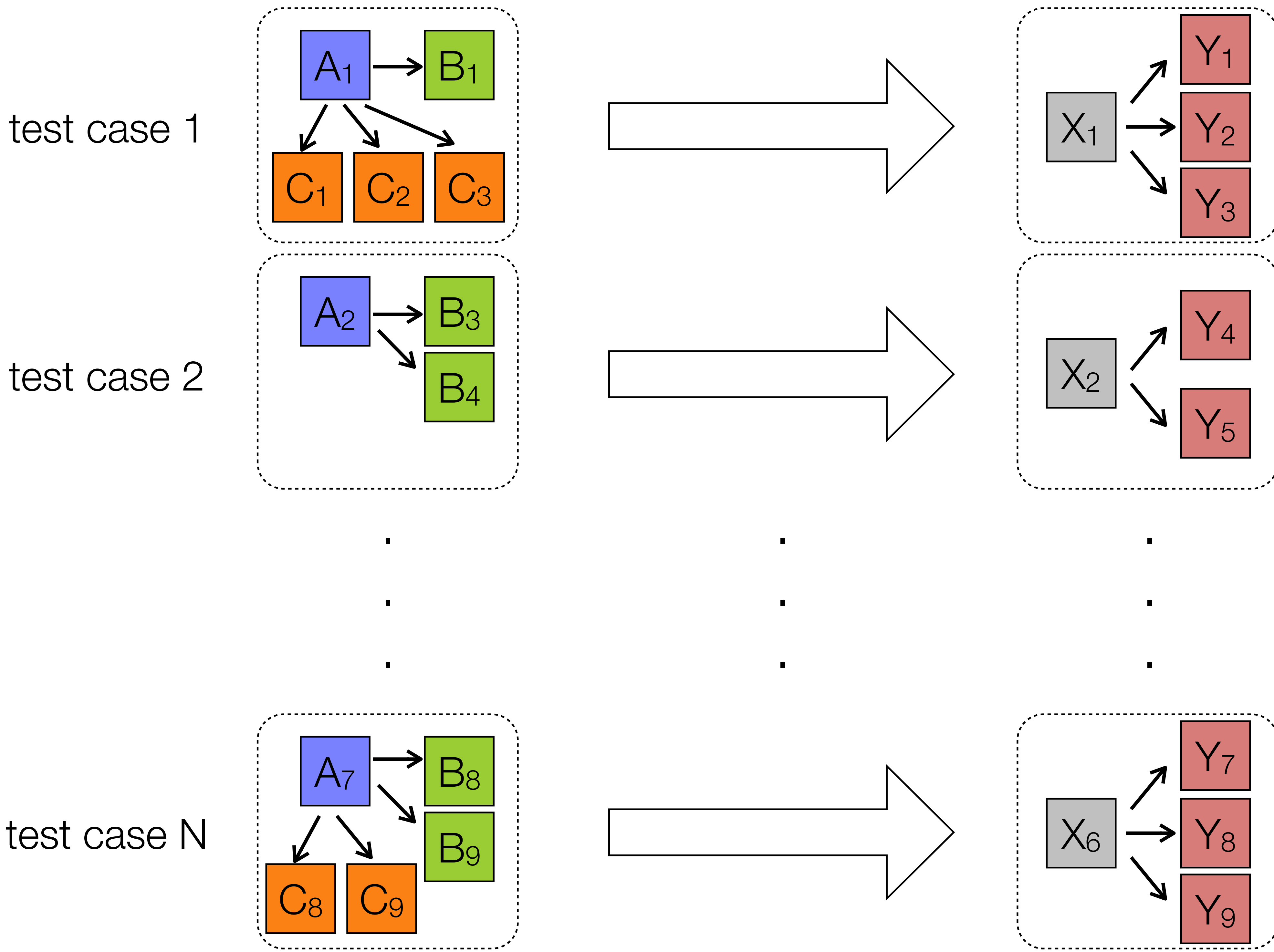
Model Transformations

A common operation in MDE toolchains is the transformation of a model to another that conforms to a different metamodel



Testing Model Transformations

To test a transformations requires a set of (possibly random) test cases in which the input data is a model instance



Motivation

Our case study is motivated by a project in which model of behaviour is transformed into a form than can be embodied on a Lego Mindstorms robot

http://lego.wikia.com/wiki/File:LEGO_Mindstorms_NXT.jpg.jpeg

Problem Statement

How can we randomly generate models for testing transformations so that a small test set of the models satisfies our testing objective?

Context

Approach

Implementation

Case Study

Context-Free Grammars

A Context-Free Grammar can be used to specify how to construct ‘well-formed’ test data

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} \\
\mathrm{Num} &\rightarrow \text{‘0’} \mid \text{‘1’} \mid \text{‘2’} \mid \text{‘3’} \mid \text{‘4’} \mid \text{‘5’}
\end{aligned}
$$

$4 + 2 * 5$

3

$3 * 2 - 5 / 0$

$/ \ 3 \ -$

Stochastic Grammars

By annotating productions rule with weights, a probability distribution is defined over the language defined by the grammar

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} \\
\mathrm{Num} &\rightarrow \underset{1}{\text{‘0’}} \mid \underset{3}{\text{‘1’}} \mid \underset{3}{\text{‘2’}} \mid \underset{3}{\text{‘3’}} \mid \underset{3}{\text{‘4’}} \mid \underset{3}{\text{‘5’}}
\end{aligned}
$$

Innovation: Conditional Weights

Making the weights conditional on the values of other variables introduces a limited form of context-sensitivity

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} \\
\mathrm{Num} &\rightarrow \text{‘0’} \mid \text{‘1’} \mid \text{‘2’} \mid \text{‘3’} \mid \text{‘4’} \mid \text{‘5’}
\end{aligned}
$$

‘+’	2	2	2	2	2	2
‘-’	2	2	2	2	2	2
‘*’	2	2	2	2	2	2
‘/’	1	3	3	3	3	3

Innovation: Binned Scalars

Adaptively ‘binning’ scalar variables enables a compact representation of distributions over large intervals

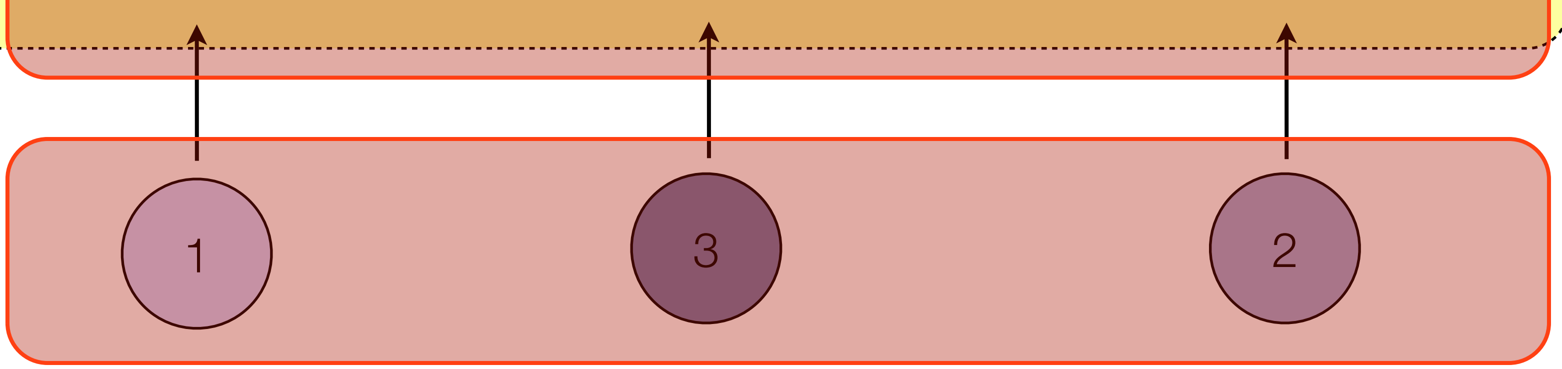
$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} \\
\mathrm{Num} &\rightarrow \text{‘0’} \mid \text{‘1’} \mid \text{‘2’} \mid \text{‘3’} \mid \text{‘4’} \mid \text{‘5’}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘/’} \\
\mathrm{Num} &\rightarrow \text{‘[0,228]’} \mid \text{‘[229,433]’} \mid \text{‘[434,511]’}
\end{aligned}
$$

Metaheuristic Search

To optimise a distribution, search is applied to the weights, the conditionality between variables, and the partitioning of scalar ranges

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{Expr} \\
\mathrm{Expr} &\rightarrow \mathrm{Num} \mid \mathrm{Expr\ Op\ Expr} \\
\mathrm{Op} &\rightarrow \text{`+'} \mid \text{`-'} \mid \text{`*'} \mid \text{`/'} \\
\mathrm{Num} &\rightarrow \text{`[0,228]'} \mid \text{`[229,433]'} \mid \text{`[434,511]'}
\end{aligned}
$$



Context

Approach

Implementation

Case Study

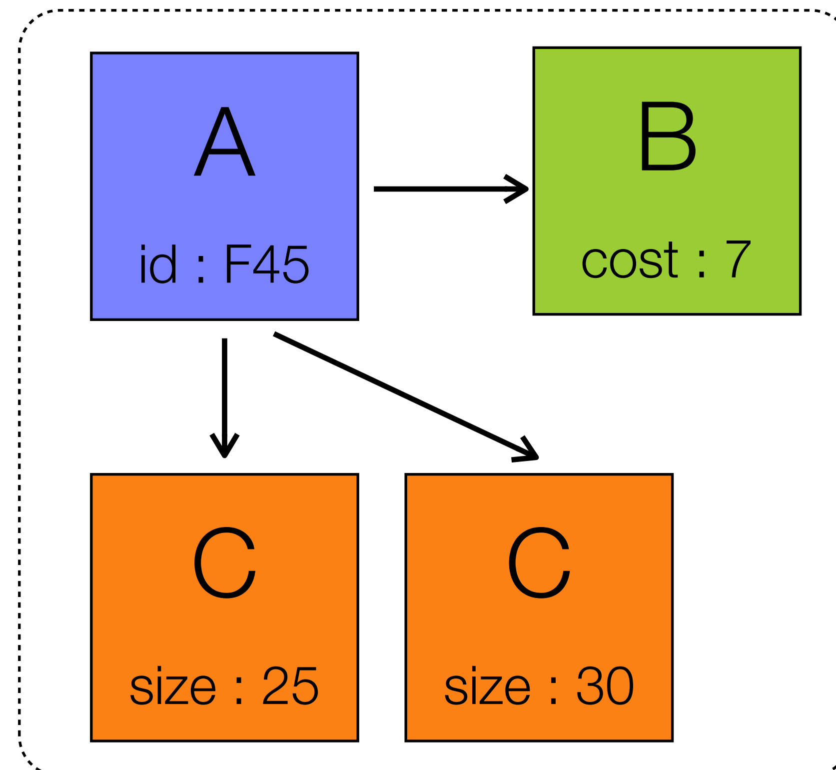
HUTN

HUTN is a textual notation for model instances

Model Instance

```
A {
  id: “F45”
  b: B { cost: 7}
  c: C { size: 25}, C {size: 30}
}
```

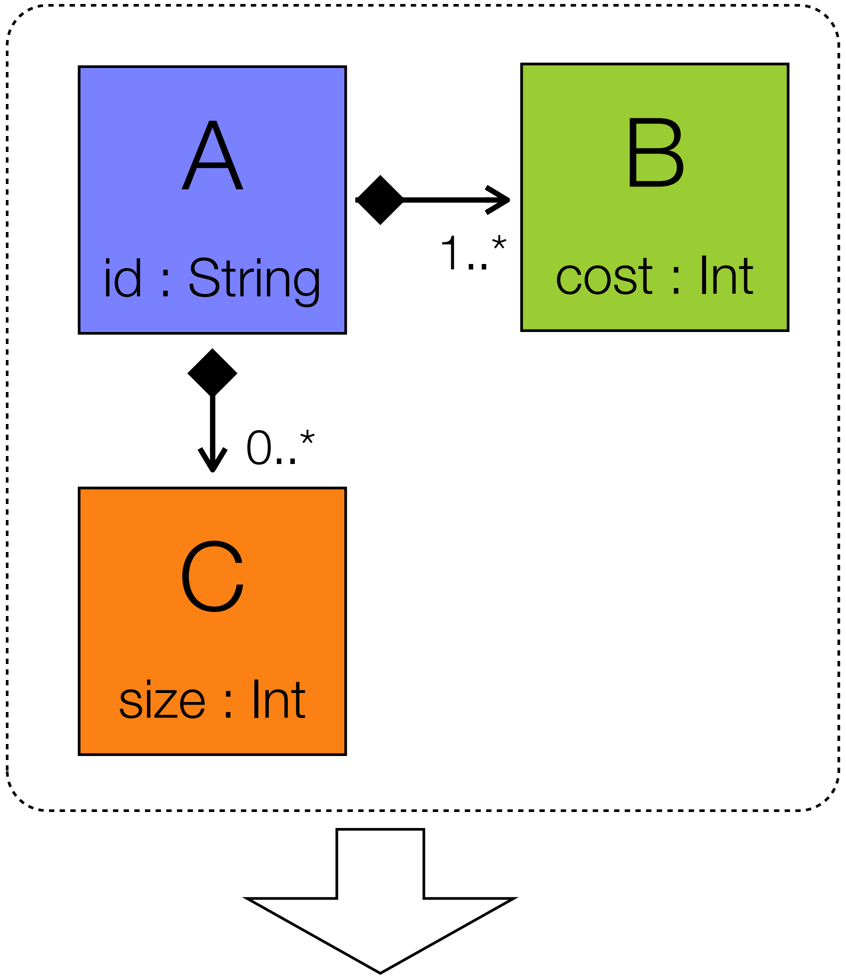
Model Instance



Grammar To Emit HUTN

We use a stochastic context-free grammar that emits HUTN which complies with the chosen metamodel

Metamodel

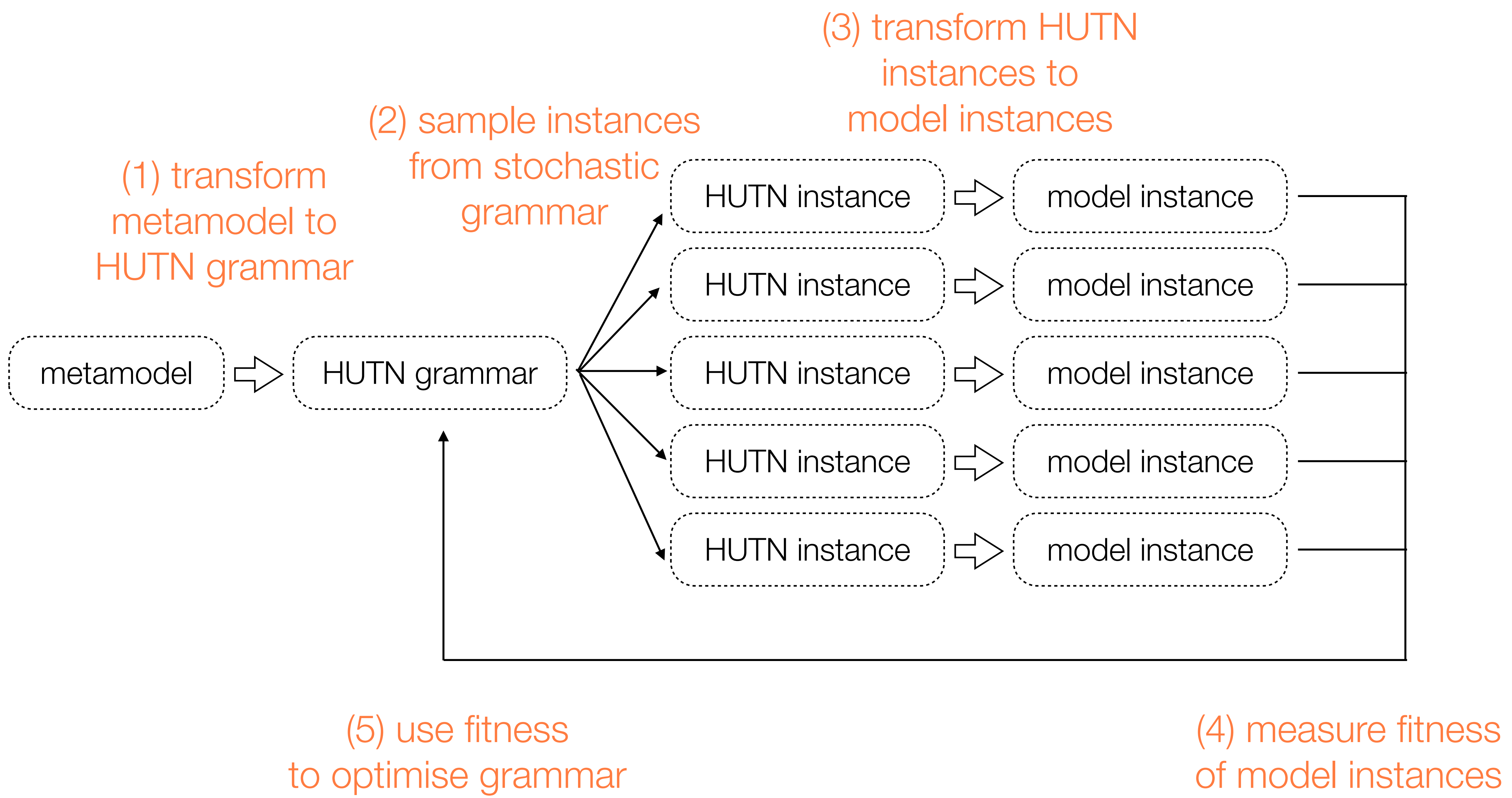


HUTN Grammar

$$
\begin{aligned}
S &\rightarrow A \\
A &\rightarrow \text{`A' `\{' `id:' String `b:' B1..* `c:' C0..* `\}'} \\
B1..* &\rightarrow B \mid B \text{ `,' } B1..* \\
B &\rightarrow \text{`\{' `cost:' Cost `\}'} \\
Cost &\rightarrow \text{`[0,100]'} \\
\ldots &\rightarrow \ldots
\end{aligned}
$$

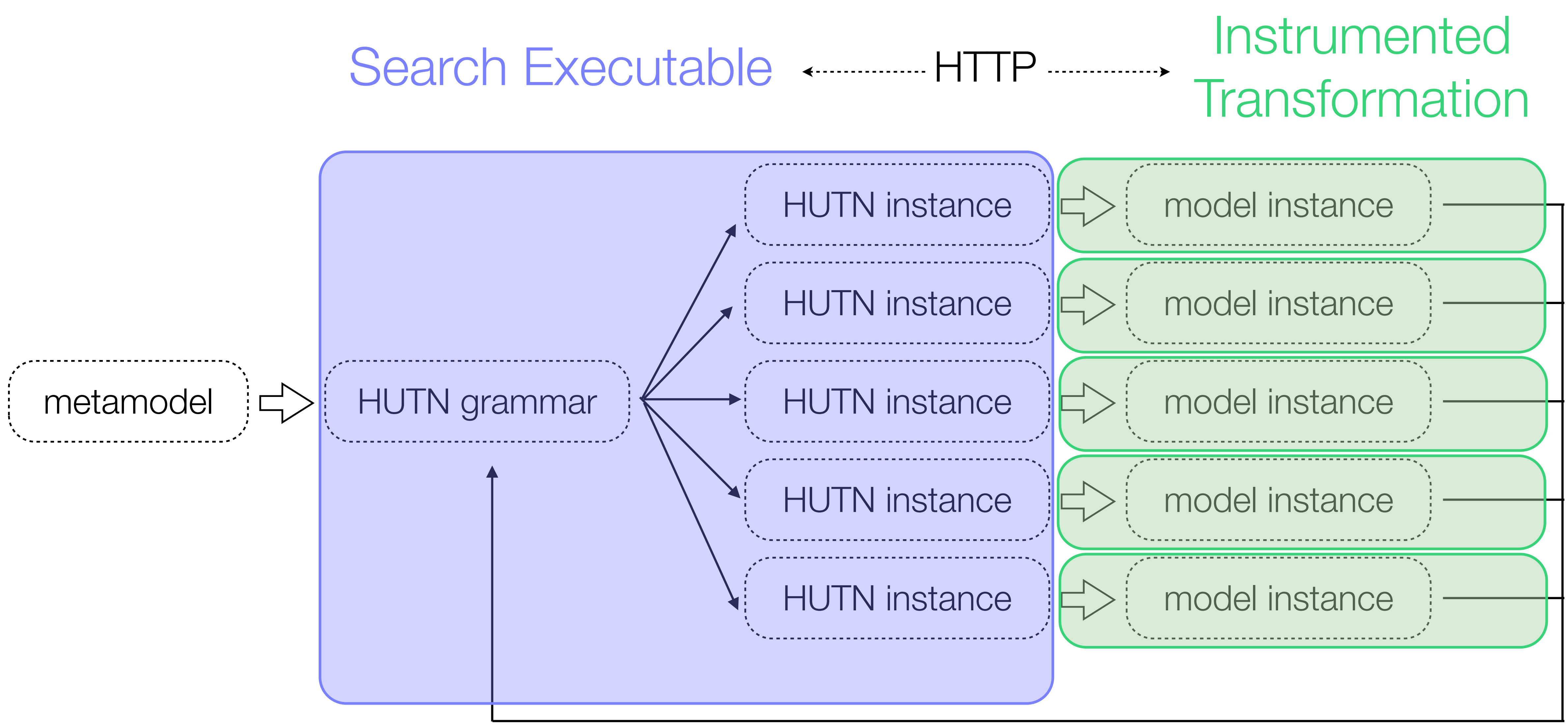
Optimisation Process

The HUTN grammar is optimised by evaluating set of models sampled from the candidate grammar



Physical Implementation

Search executable optimises and samples from the grammar; in a servlet, instrumented MDE transformation converts HUTN to instances and assesses; components communicate over HTTP



Context

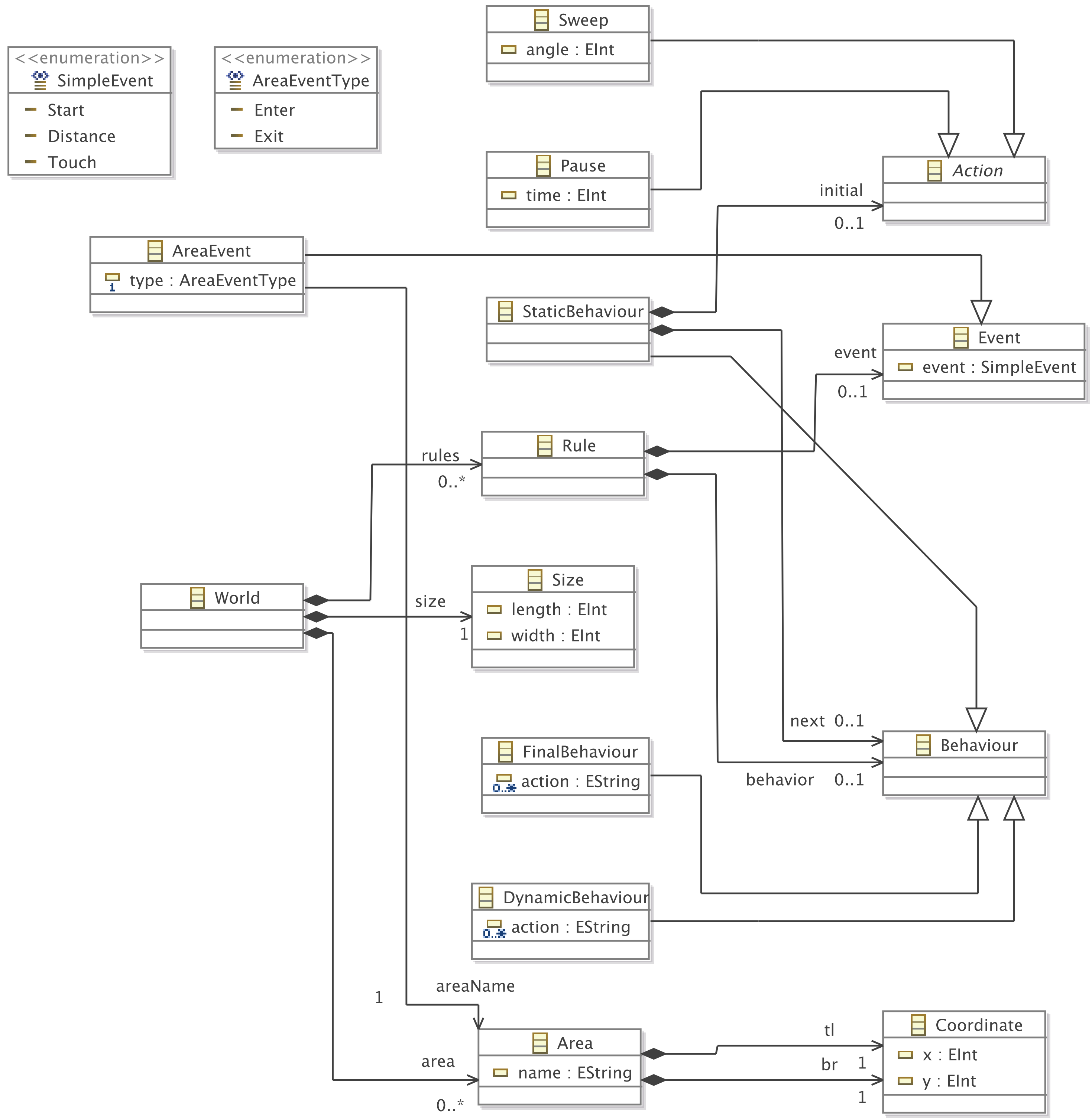
Approach

Implementation

Case Study

Source Metamodel

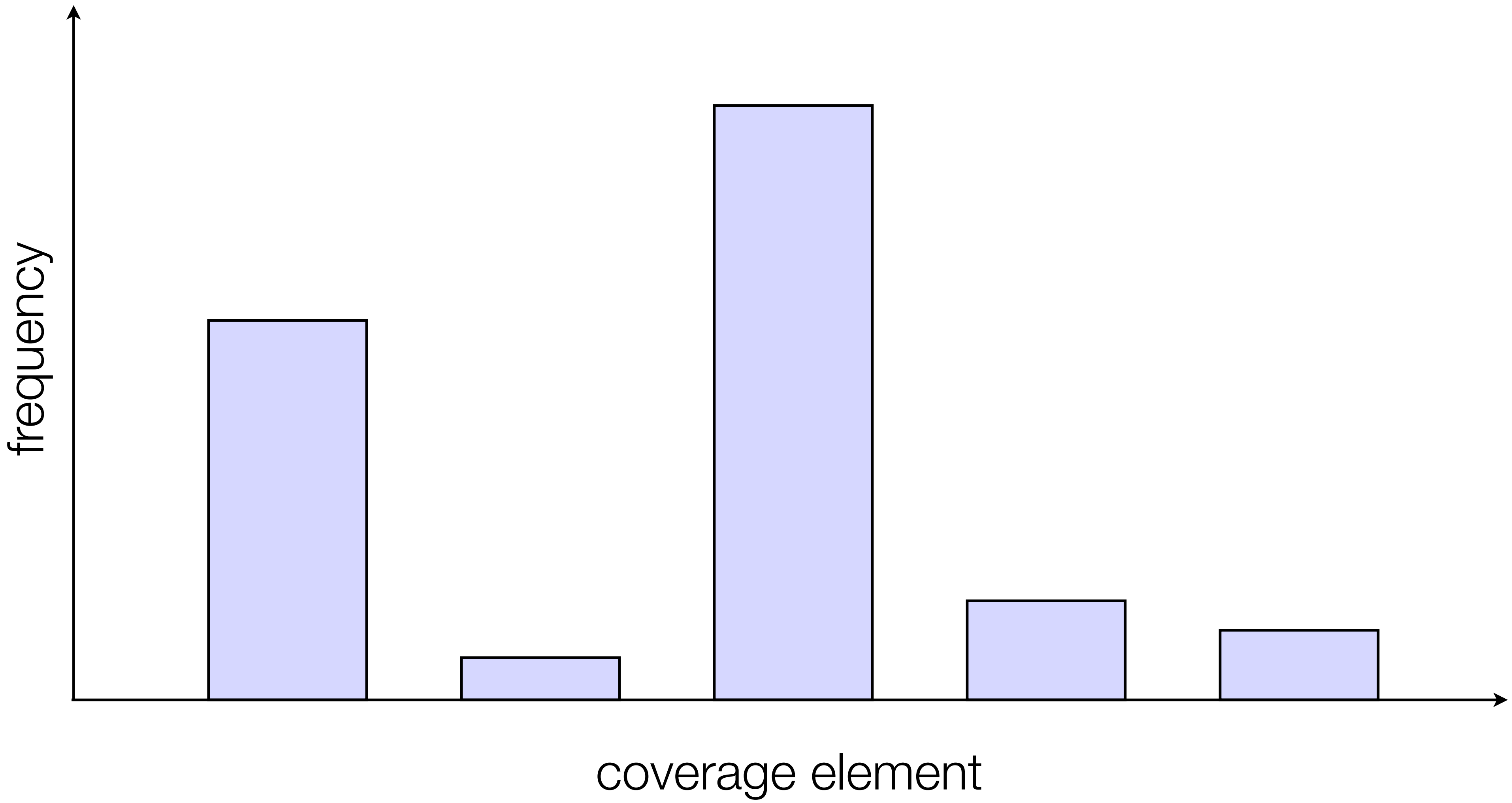
Specifies the models that are the inputs to the transformation under test



http://lego.wikia.com/wiki/File:LEGO_Mindstorms_NXT.jpg.jpeg

Objective: Statistical Testing

Want to maximise frequency of covering every rule, guard, and condition as frequently as possible in order to minimise test set size



Experiments

Compare efficiency of optimised and unoptimised grammar; random search as measure of 'difficulty'

Optimised using hill-climbing (800 evaluations)

V

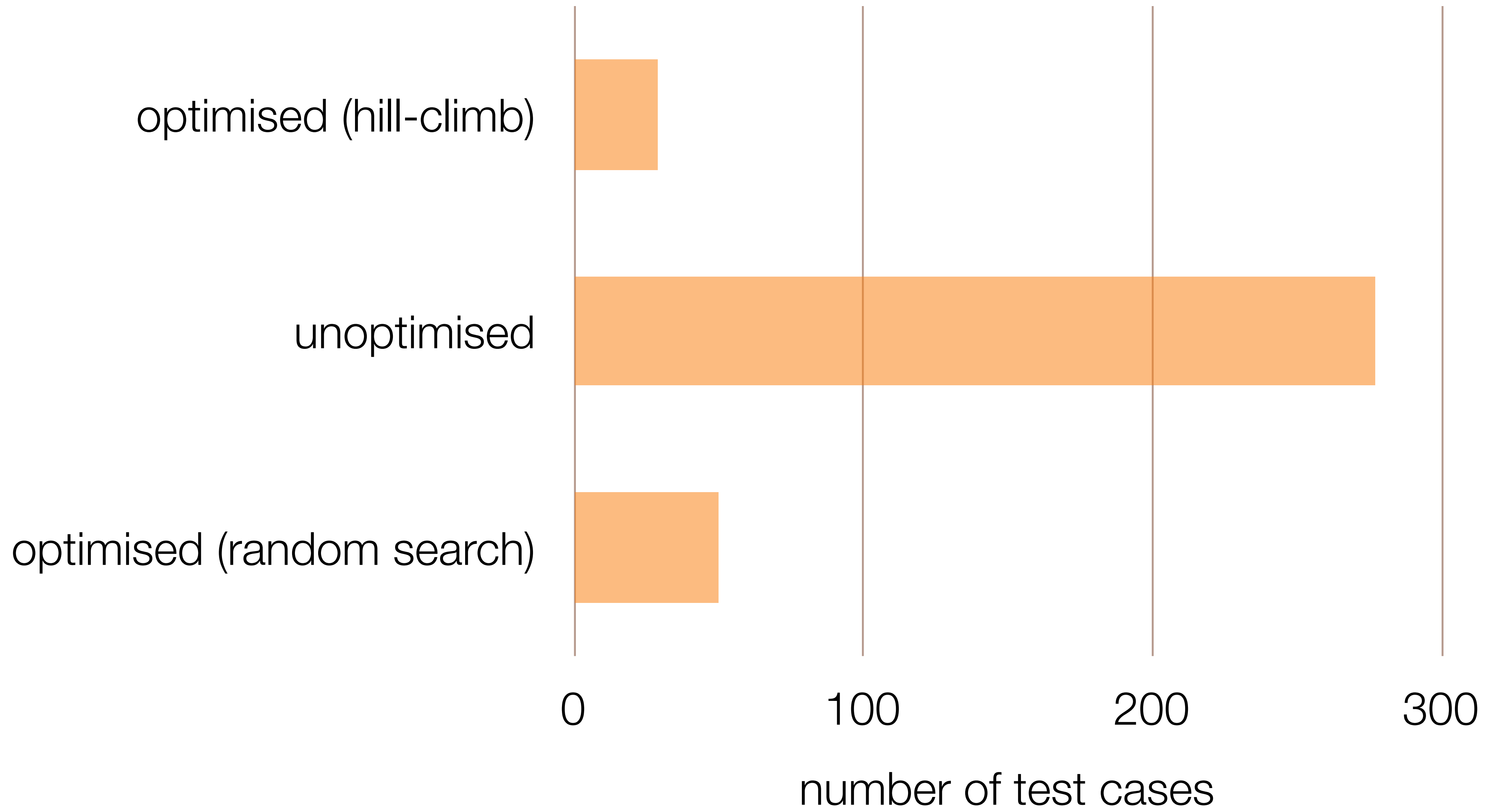
Unoptimised ('uniform' distribution)

V

Optimised using random search (800 evaluations)

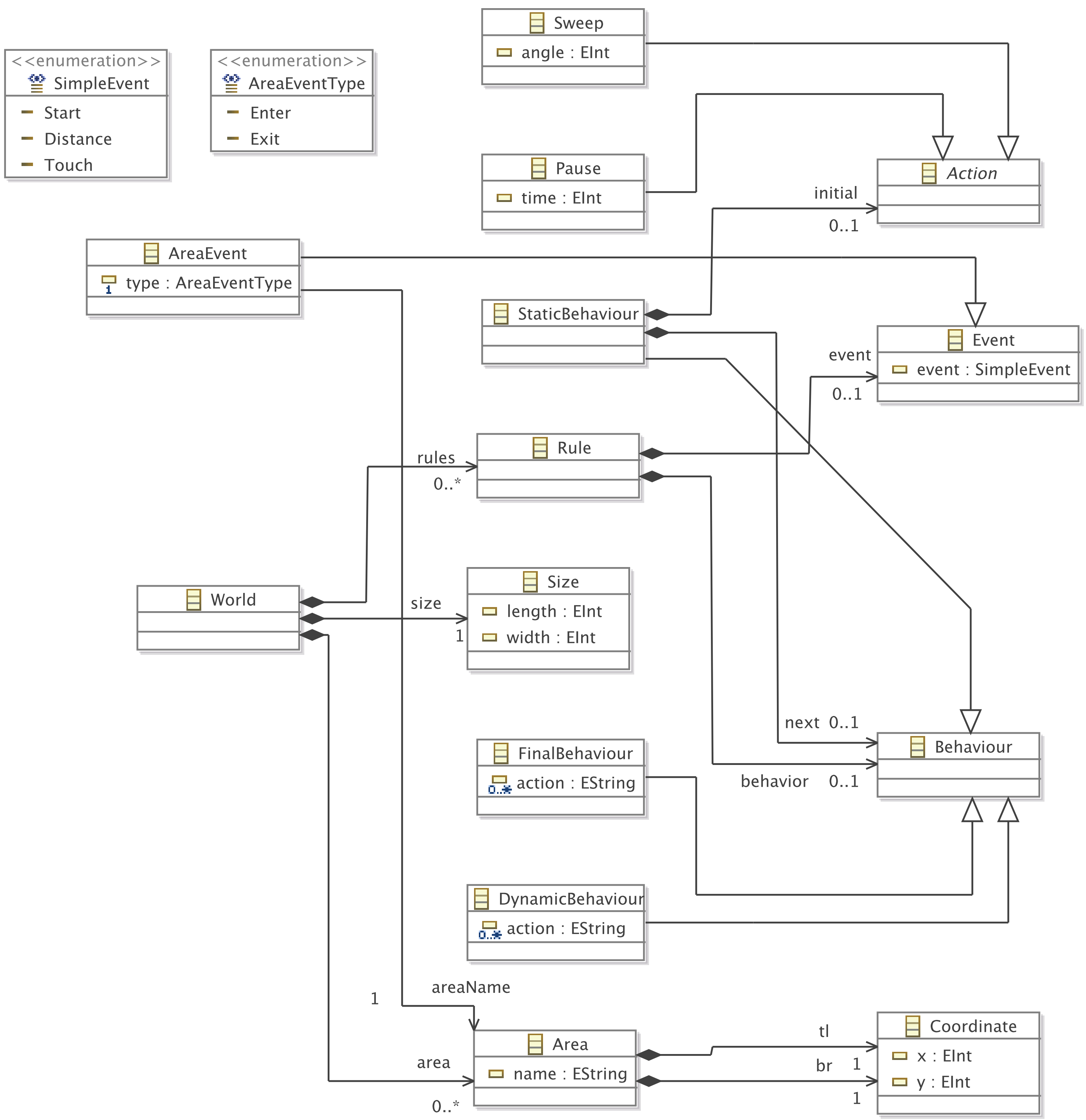
Results

Number of test cases to cover all elements
(with a 90% likelihood) - smaller is better



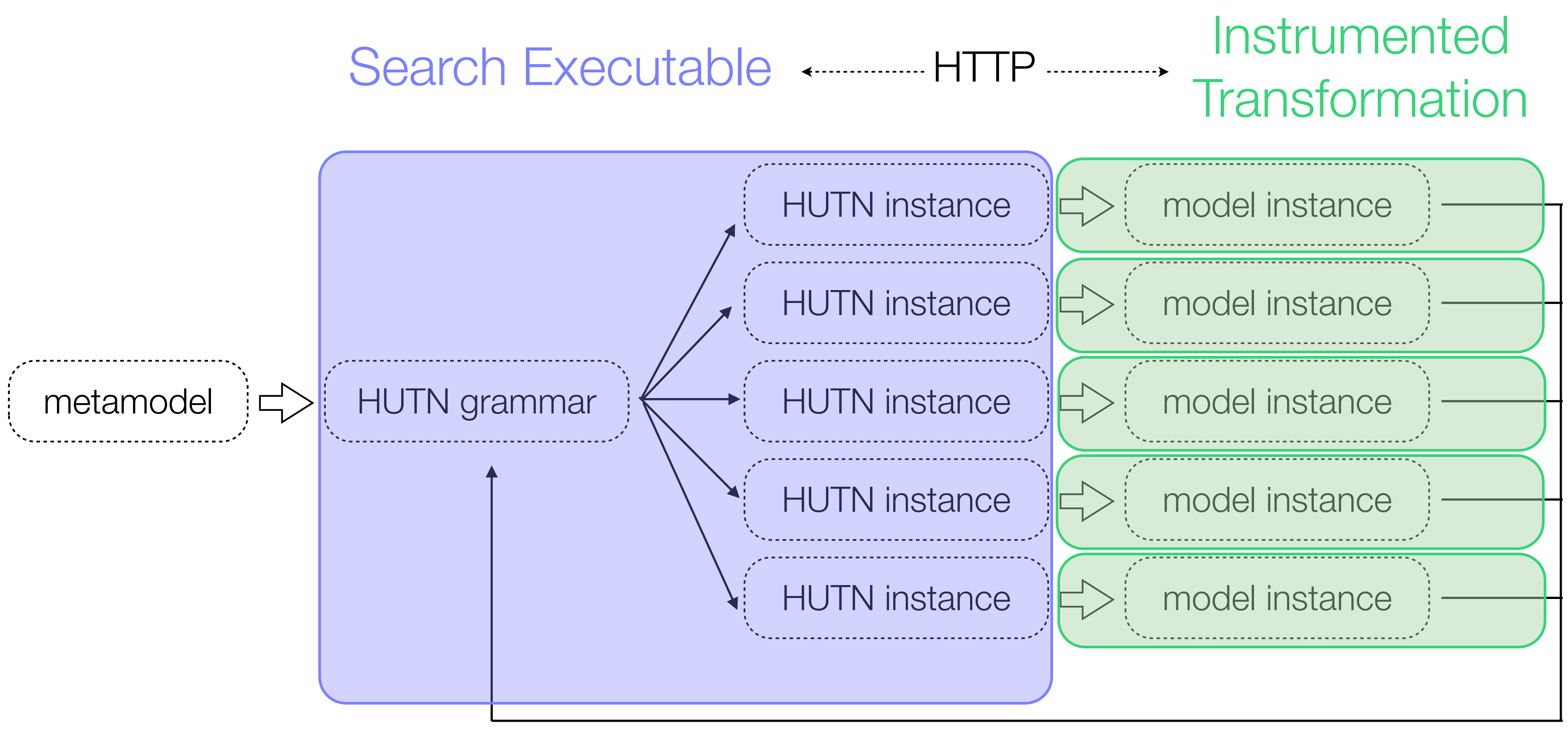
Other Outcomes

Process highlighted ambiguities and missing information in the original metamodel



Future Work

Automate metamodel to HUTN grammar conversion; speed up evaluation



Further Details

Simon Poulding, Robert Alexander, John A. Clark, and Mark J. Hadley
The Optimisation of Stochastic Grammars to Enable Cost-Effective Probabilistic Structural Testing
Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2013)
(to appear)

Louis M. Rose and Simon Poulding
Efficient Probabilistic Testing of Model Transformations using Search
Proceedings of 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMBSE 2013)
(to appear)