Pirates Are Watching the Clouds

Ethan Jackson, Nikolaj Bjørner and Wolfram Schulte
Research in Software Engineering (RiSE), Microsoft Research

Dirk Seifert, Markus Dahlweid and Thomas Santen,
European Microsoft Innovation Center (EMIC), Aachen, Germany

1. Introduction

http://research.microsoft.com/formula

http://research.microsoft.com/~ejackson/wicsa2011

Big Systems Need Formal Methods

Unlike a single algorithm/method we want to understand system-level properties that are hard to reason about using only code.

Ex 1: Users want to run their apps in the cloud.
Problem: The data center needs to find deployments in the presence of constraints .
Solution: **Model apps and constraints** and synthesize deployments.

Ex 2: Integrated web-services that use each others data to make decisions.
Problem: The pirates want to know if they can exploit the decision process.
Solution: **Model how decisions are made** and find suspicious scenarios, so correct flaws early.

A Formal Specification Language

FORMULA is a formal specification language targeting model-based development, shaped by the following scenarios:

Declarative Specifications of:

Rule-based systems *(e.g. policy languages)*
Good Configurations of Complex Systems *(e.g. clouds)*
Classes of Software Architectures *(e.g. embedded systems)*
Deployment Problems of Architectures to Compute Fabrics *(e.g. schedulable deployments)*

Automated Reasoning on Specifications for:

Design space exploration,
Bounded symbolic model checking,
Test-case Generation,
Consistency Checking

Core: ADTs + Open World CLP

Algebraic data types
plus novel regular type inference for representing abstractions.

Strongly Typed Bottom-up Constraint Logic Programming
for describing the logic of the abstraction.

Open World Semantics by Efficient Symbolic Execution to SMT
to find diverse solutions to the problem.

Module System with Formal Composition Operators
for safely building complex specifications.

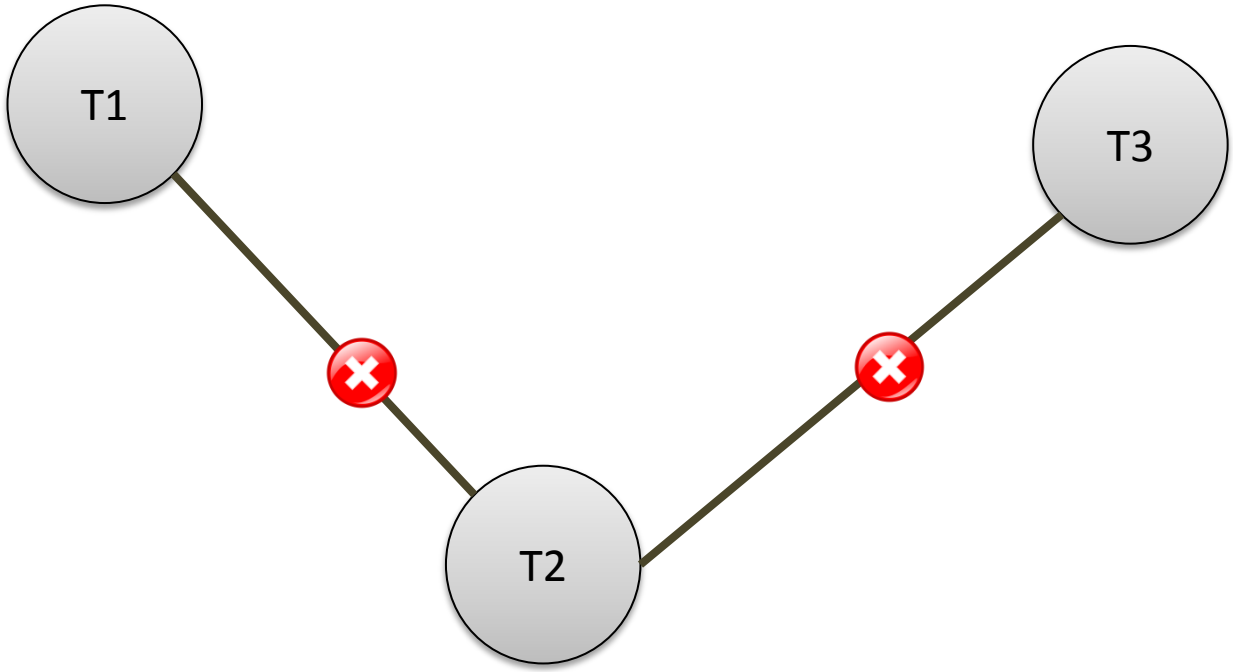
2. Deploying to a Cloud

http://research.microsoft.com/formula

http://research.microsoft.com/~ejackson/wicsa2011

Applications (I)

For simplicity, assume an application is just a task.
Two tasks can be in conflict, meaning they should not execute on the same node.



Applications (II)

The "domain" keyword starts and abstraction

Data type constructors with labeled arguments and type constraints

```
domain Applications
{
        App ::= (id: String).
        [Closed]
        Conflict ::= (t1: App, t2: App).
}
```

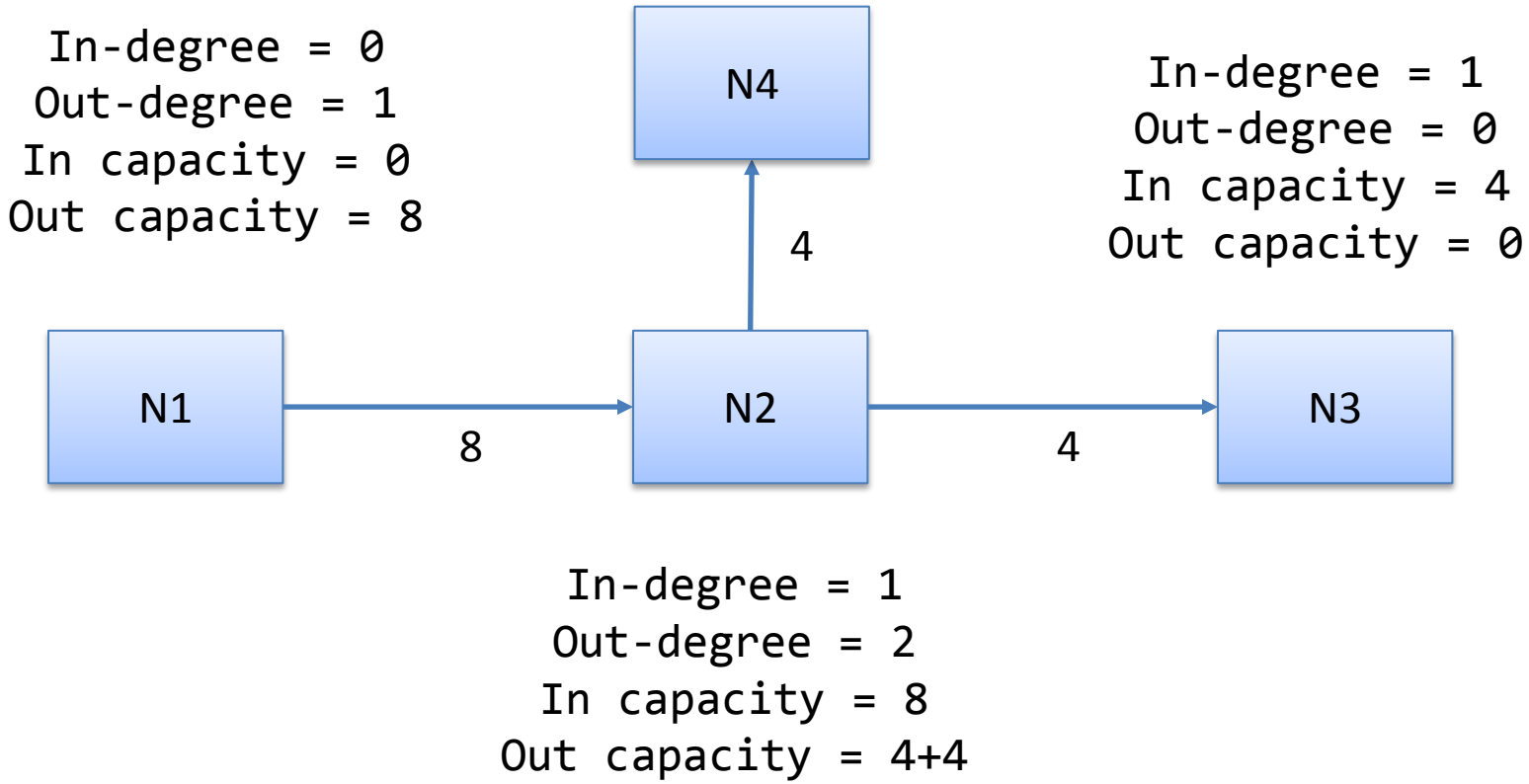
A "model" is claim of conformance

And a set terms built using data type constructors

```
model ApplicationModel of Applications
{
        t1 is App("HBI Database")
        t2 is App("Web Server")
        t3 is App("Voice Recognition")
        Conflict(t1, t2)
        Conflict(t2, t3)
}
```

The Cloud (I)

Nodes are connected by channels with communication capacities.
No node can support more than two incoming and outgoing channels.
Capacities must be balanced on node with incoming and outgoing channels.



The Cloud (II)

Special annotations for common constraints

Named "queries" can be treated like Boolean variables.

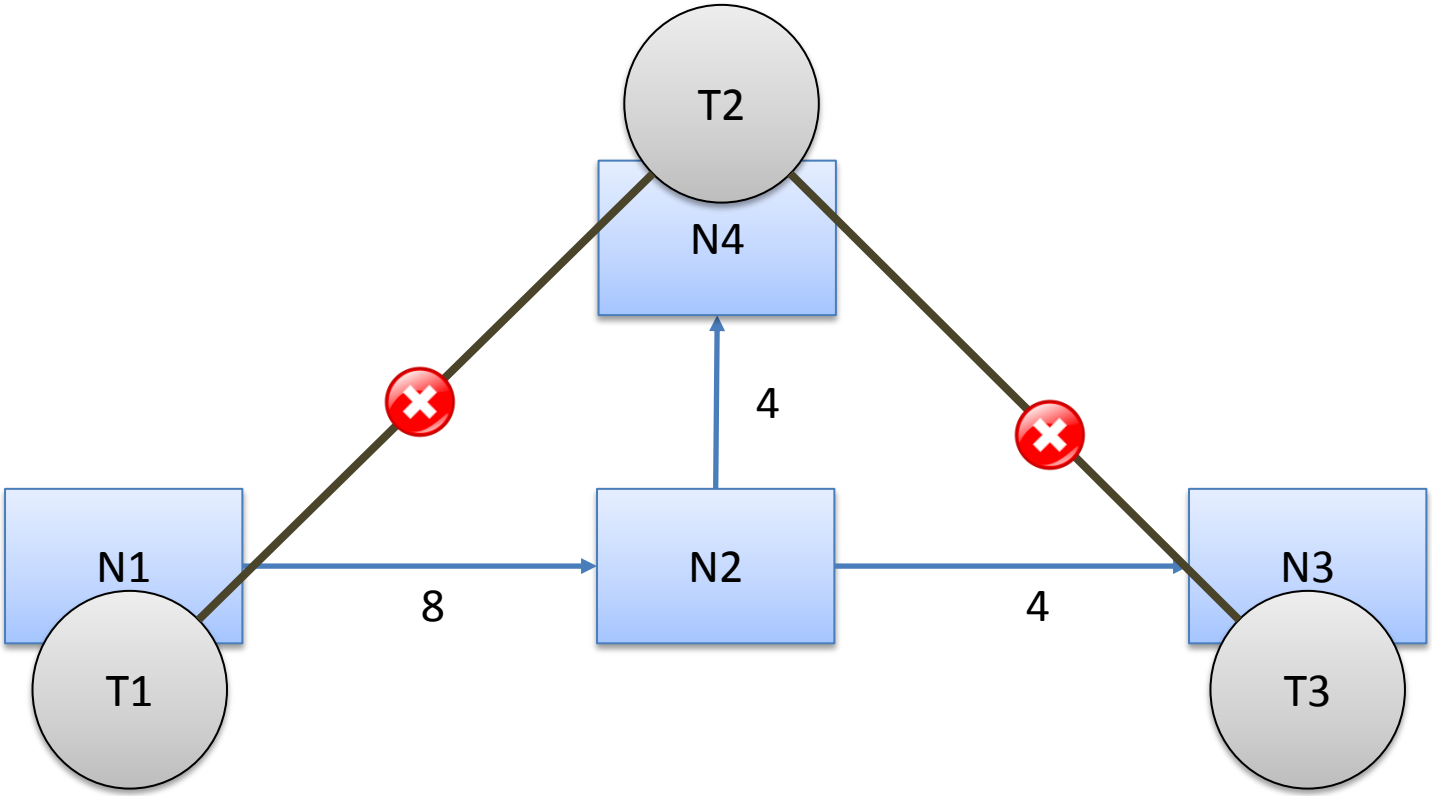
Rules derive complex information

The "conforms" query determines the models.

```
domain Cloud {
  Node    ::= (id: Integer).
  [Closed(fromNode, toNode)]
  [Unique(fromNode, toNode -> cap)]
  Channel ::= (fromNode: Node, toNode: Node,
               cap: PosInteger).

  bigFanIn  := n is Node, count(Channel(_,n,_)) > 2.
  bigFanOut := n is Node, count(Channel(n,_,_)) > 2.

  mustBal(n) :- Channel(_,n,_), Channel(n,_,_).

  clog := mustBal(n),
          sum(Channel(_,n,_),2) !=
          sum(Channel(n,_,_),2).

  conforms := !(bigFanIn | bigFanOut | clog).
}
```

Deployments (I)

Tasks should be place on nodes so all conflict constraints are respected.



Deployments (II)

The "extends" keyword safely composes

Only need to write the new constraints.

```
domain Deployment extends Applications, Cloud
{
  [Closed] [Function(fromApp -> toNode)]
  Binding ::= (fromApp: App, toNode: Node).

  inConflict := Binding(t1, n), Binding(t2, n),
                Conflict(t1, t2).

  conforms   := !inConflict.
}
```

Simplified example, but not an easy one:

A coloring problem,
A forbidden-subgraph problem
Linear arithmetic problems

Realistic problems contain constraints like these.

Solve in Any Direction

The user constructs a partial model to represent the degrees of freedom in the problem. Degrees of freedom can be anywhere.

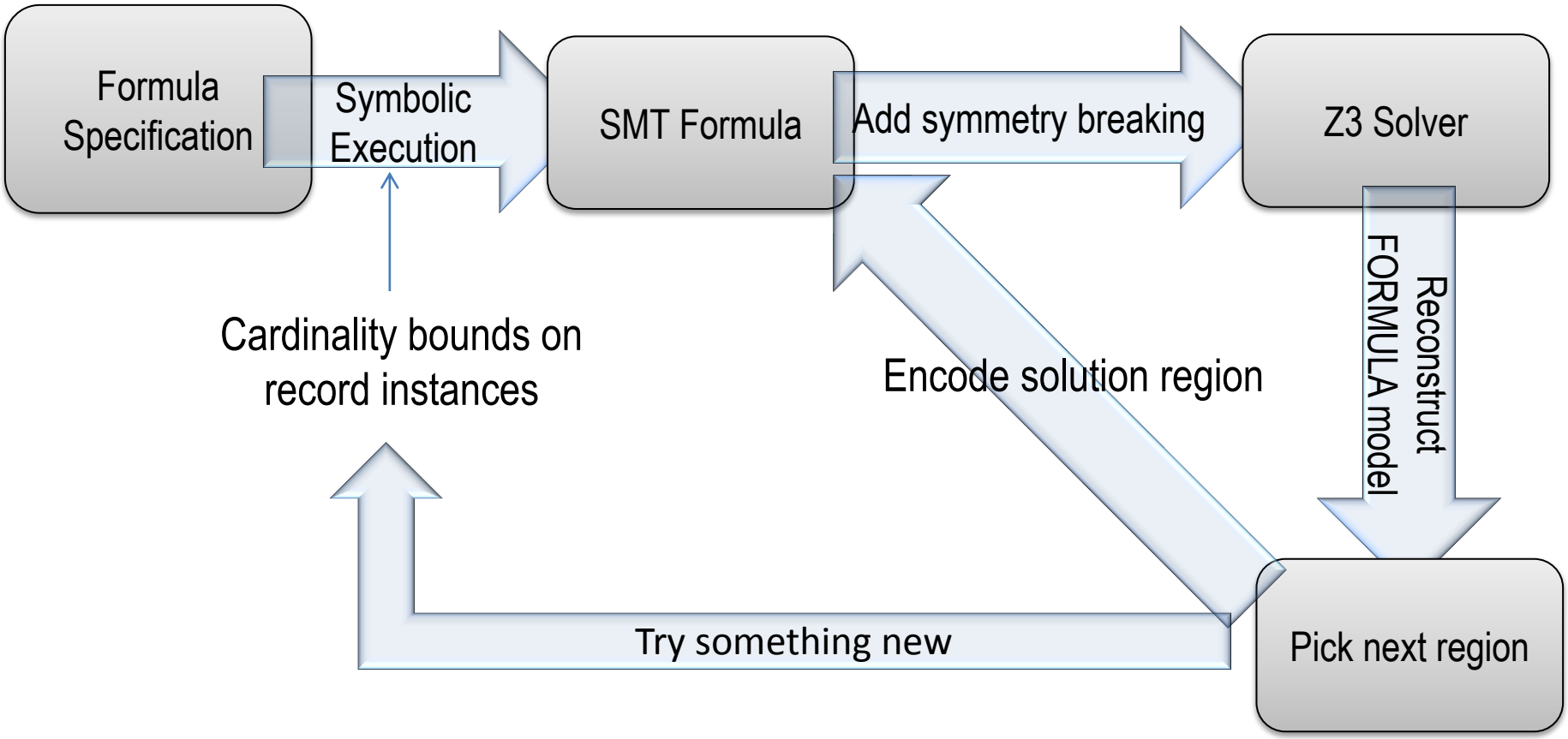
Entities that must be in any solution

Explicit degrees of freedom. There are also implicit degrees of freedom, like binding.

```
partial model Ex of Deployment
{
  t1 is App("HBI Database")
  t2 is App("Web Server")
  t3 is App("Voice Recognition")
  Conflict(t1, t2)
  Conflict(t2, t3)

  n1 is Node(1)
  n2 is Node(2)
  n3 is Node(3)
  c1 is Channel c2 is Channel c3 is Channel
  c4 is Channel c5 is Channel c6 is Channel
  c7 is Channel c8 is Channel c9 is Channel
}
```

Design Space Exploration

Given a spec and a partial model, then symbolic execution constructs a formula representing the design space.



3. Pirates and Loop-Holes

http://research.microsoft.com/formula

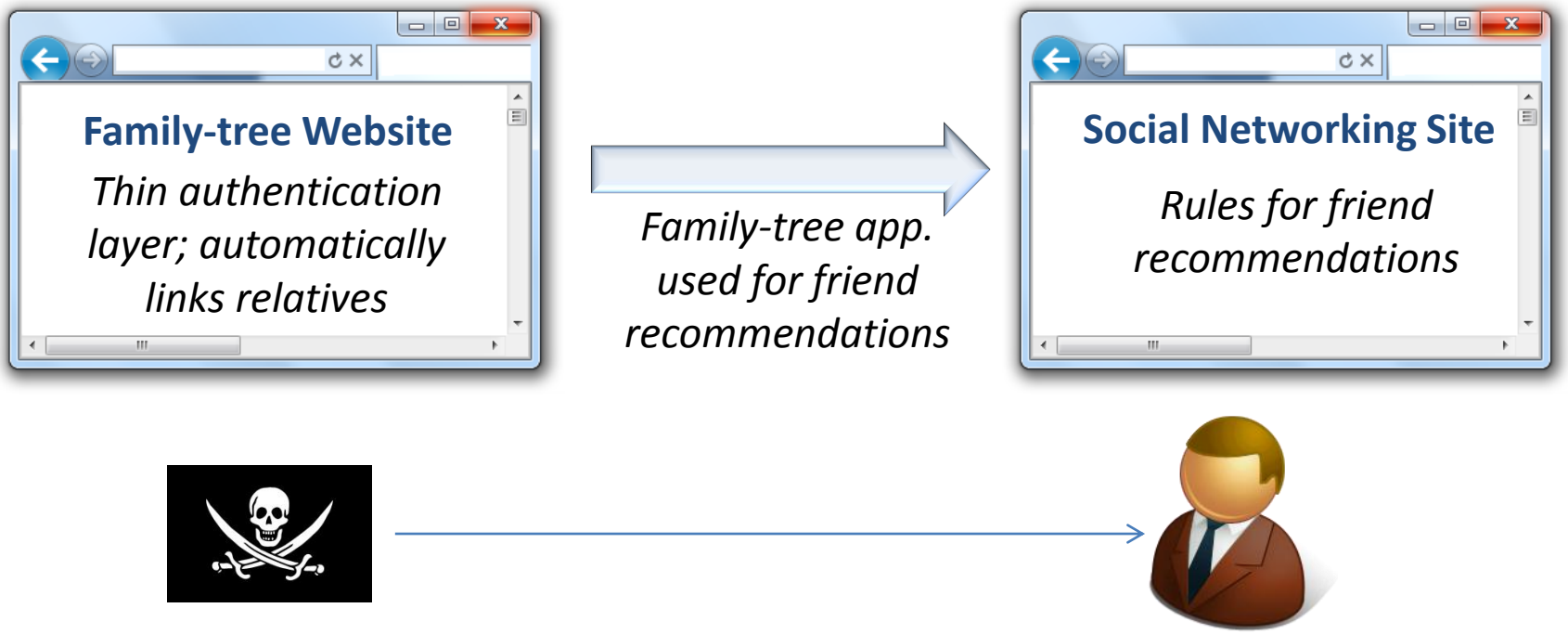
http://research.microsoft.com/~ejackson/wicsa2011

Integration of Web Services

A social networking site has rules for friend recommendations

But integrates data from other sites, importing their rules into its trust boundary.

The pirate uses this composition to try to gain your trust.



Rules of the Social Network

```
domain Principals
{
   Person ::= (first: String, last: String).
}

domain SocNetwork extends Principals
{
  [Closed]
  Friend         ::= (Person, Person).
  isDirectFriend ::= (Person, Person).
  isFofF         ::= (Person, Person).

  isDirectFriend(p1, p2),
  isDirectFriend(p2, p1) :- Friend(p1, p2).
  isFofF(p1, p2)         :- isDirectFriend(p1, p2).
  isFofF(p1, p2)         :- isFofF(p1, p), isFofF(p, p2).
  recFriend(p1, p2)      :- isFofF(p1, p2), p1 != p2,
                            fail isDirectFriend(p1, p2).

  recAndNotFofF := recFriend(p1, p2), p1 != p2, fail isFofF(p1, p2).
}
```

Can Eve Do Anything Suspicious?

Maybe she can even make up her last name…

```
partial model Net of SocNetwork
{
   pEve   is Person("Eve", _)
   pBob   is Person("Bob", "Bob")
   pChuck is Person("Chuck", "Chuck")
   pAlice is Person("Alice", "Alice")
   Friend(pAlice, pBob)
   Friend(pBob, pChuck)
}
```

The Family Tree Website

```
domain FamilyTree extends Principals
{
   [Closed]
   Database  ::= (Person).
   isRelated ::= (Person, Person).

   isRelated(p1, p2) :- Database(p1), Database(p2),
                        p1.last = p2.last, p1 != p2.
}
```

The Integration

```
domain Integration extends SocNetwork, FamilyTree
{
   recFriend(p1, p2) :- isRelated(p1, p2).
}


[Introduce(Database, 4)]
partial model NetInt of Integration
{
   pEve   is Person("Eve", _)
   pBob   is Person("Bob", "Bob")
   pChuck is Person("Chuck", "Chuck")
   pAlice is Person("Alice", "Alice")
   Friend(pAlice, pBob)
   Friend(pBob, pChuck)
}
```

A Suspicious Scenario

Not a bug in the usual sense, but a scenario that should be carefully considered.

```
model NetInt_1 of Integration at "../WICSAExamples.4ml"
{
   Person("Alice","Alice")
   Person("Bob","Bob")
   Person("Chuck","Chuck")
   Person("Eve","Chuck")

   Friend(Person("Alice","Alice"),Person("Bob","Bob"))
   Friend(Person("Bob","Bob"),Person("Chuck","Chuck"))

   Database(Person("Bob","Bob"))
   Database(Person("Chuck","Chuck"))
   Database(Person("Eve","Chuck"))
}
```

Some Use-Cases

Automotive Embedded Systems

Design space exploration over end-to-end assembly of components satisfying temporal and dataflow constraints (pilot with automotive industry).

Verifying Model Transformations

Transformations re-write high-level architecture, but want to verify they don't perturb correctness. What happens to correctness if we introduce triple-redundancy.

Reasoning About Policy Languages

Show how large sets of policies interact by generating configurations causes the policy to react in some way (pilot project with internal product groups).

DSE to Optimization

Use DSE loop combined with simulation-based ranking to find optimal designs w.r.t to system dynamics.

Thank you and Questions

http://research.microsoft.com/formula

http://research.microsoft.com/~ejackson/wicsa2011