

# 1. A FORMULA for Abstractions and Automated Analysis

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), Microsoft Research

**FORMULA**

Modeling Foundations.



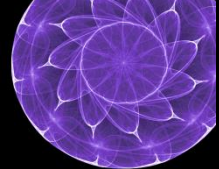
# i. Abstractions

<http://research.microsoft.com/formula>

**FORMULA**

Modeling Foundations.





# Why Abstractions?

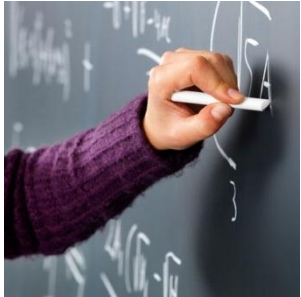
Abstractions let us formally reason about properties using only the relevant details.



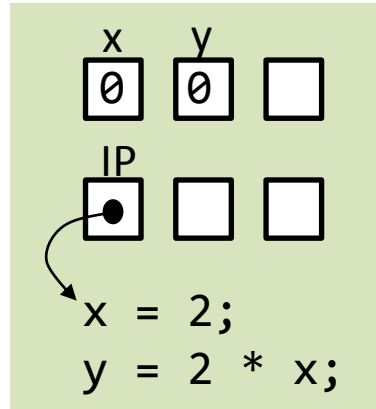
For example, we don't reason about functional properties of software with an oscilloscope...



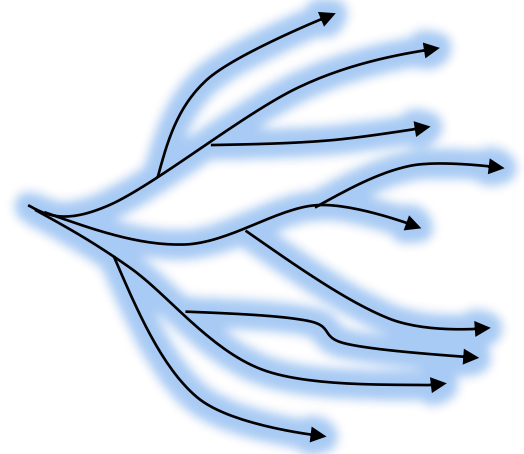
# The Right Abstraction Pays Off...



Axioms of machine



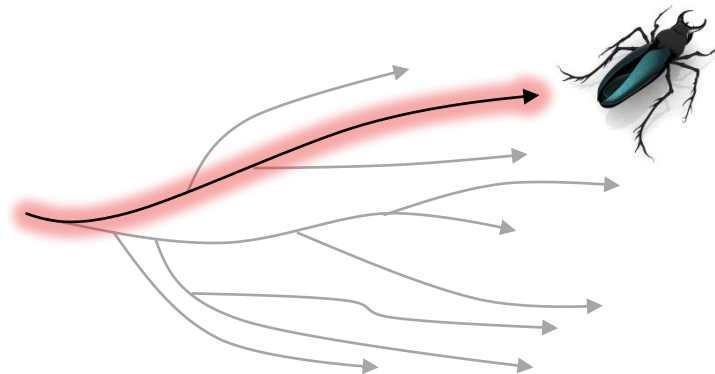
Initial Configuration



Traces

Can apply axioms to automatically reason about a **model** of a program

Trace showing a reachable bug





# Why More Than One Abstraction?

Of course, tradeoffs between precision and analyzability of programs.

Also, modern software systems have new problems where:

1. The axioms of the machine are insufficient to express the property.
2. The property is not easily phrased in terms of execution traces of programs.



# Example 1 – Data Center Configuration

How can a set of services be feasibly deployed in a data center?  
(Functional correctness of services doesn't help)

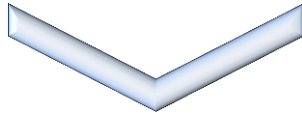
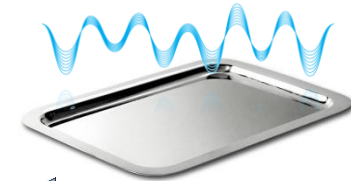
HBI Database



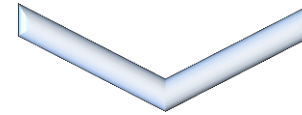
Web server



Voice rec. service



Can't do this; literally illegal in some cases.



Can't do this; may overload CPU.

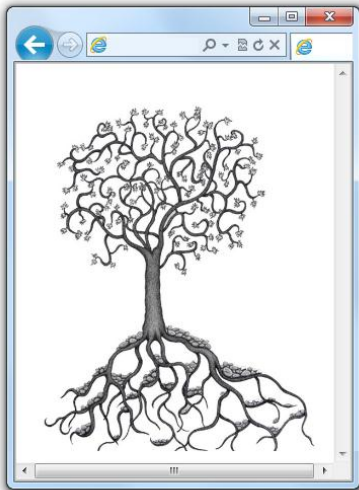
Need some model of conflicts between services, but not necessarily implementation details.



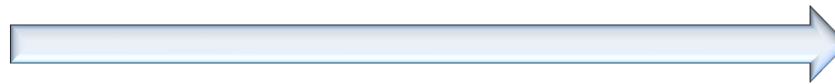
# Example 2 – Web-Service Integration

When (correct) web-services are mashed-up can new vulnerabilities arise?

Genealogy site



Genealogy data to make friend recommendations



Can Evil Eve get recommended as friend to Bob?

Social networking site



Policy for computing friend recommendations

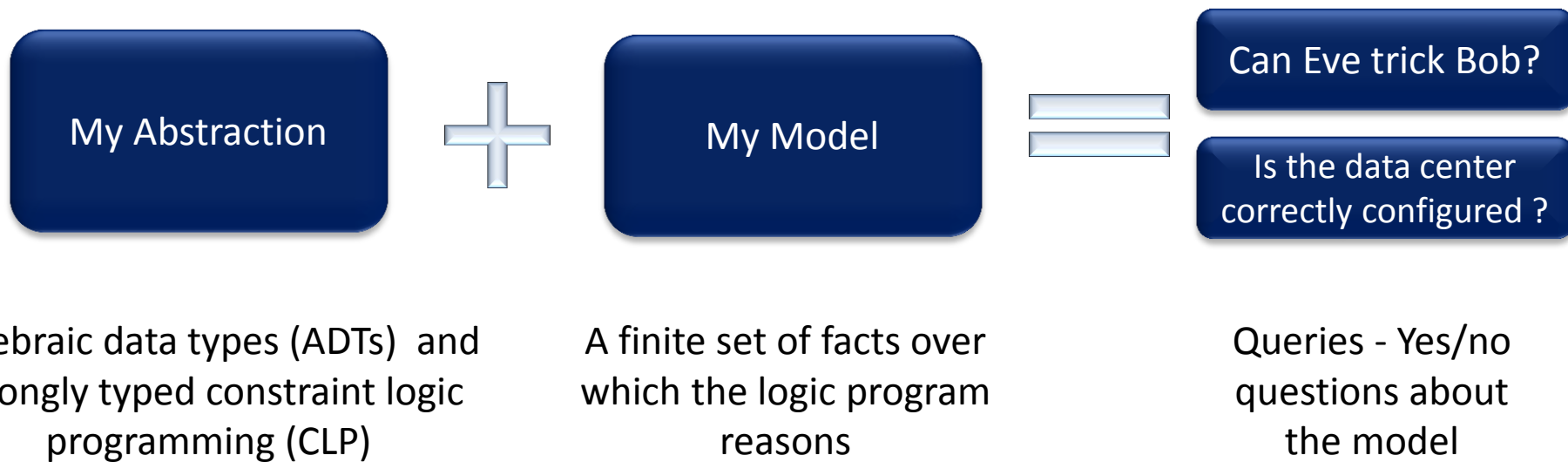
Policy for computing genealogical relationships

Full static verification would be very difficult.  
Abstracting to the data policies focuses the problem.



# A FORMULA for Abstractions

The **FORMULA language** is a general framework for specifying abstractions.

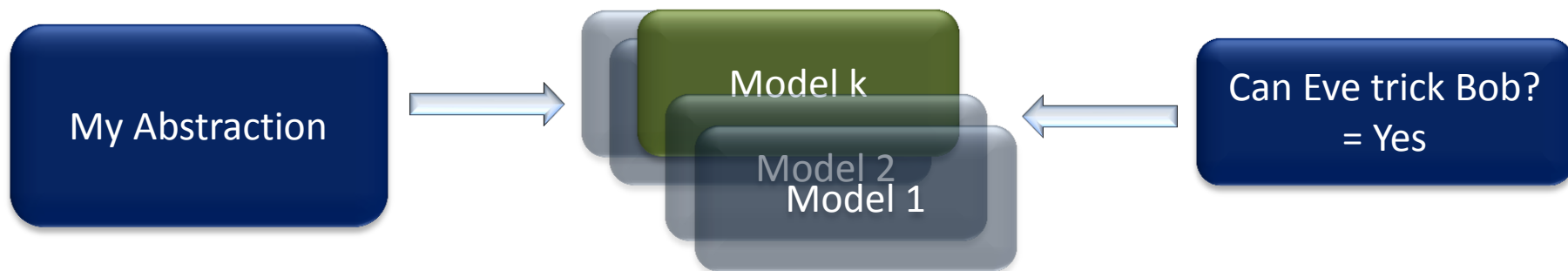


The class of CLP means axioms are a subclass of FPL over background theories



# A FORMULA for Analysis

The **FORMULA solver** provides general state-of-the-art automated analysis by **finite model finding**.



Assume the logic program does not state all the facts (open world assumption).

Search for models (worlds) where the query holds

Require a query to hold

Can encode design space exploration, bounded model checking, schedulability analysis etc... as finite model finding over OWA programs.

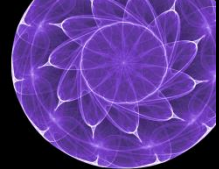
## ii. Open-World Reasoning

<http://research.microsoft.com/formula>

**FORMULA**

Modeling Foundations.



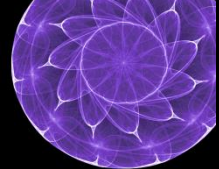


# The Closed World Assumption (CWA)

Typically, CLP semantics state a proposition is false if it is false in the **least model** of the logic program.

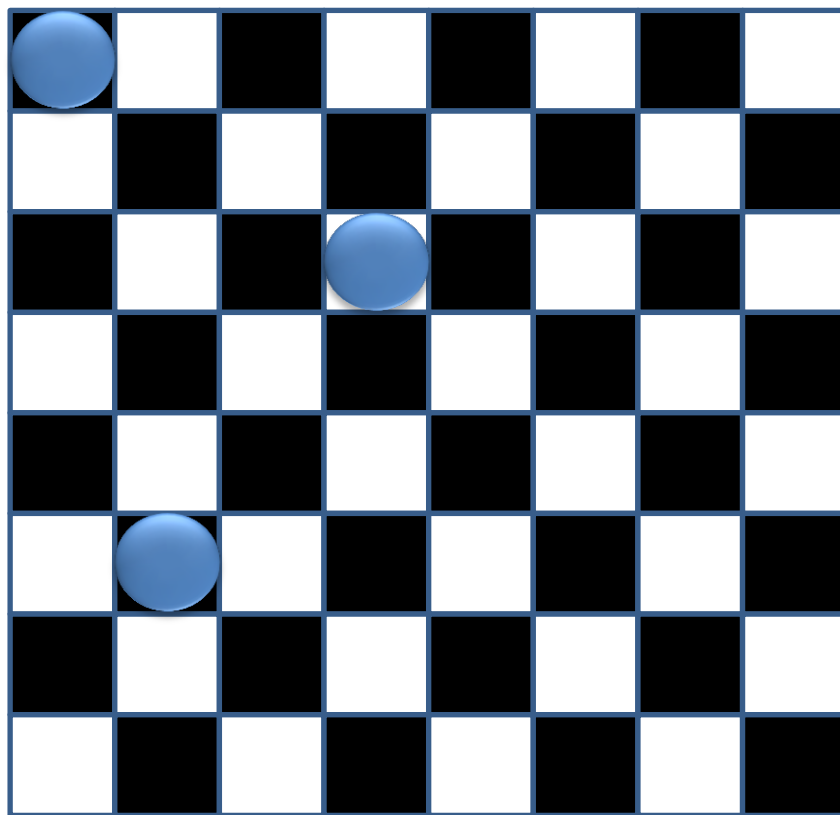
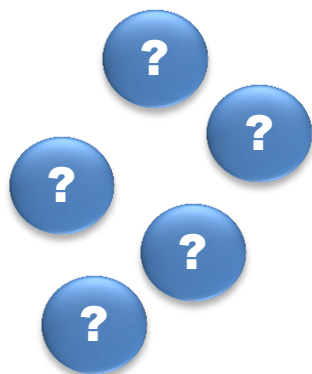
This assumption hinders our desire to search over worlds where a query holds.

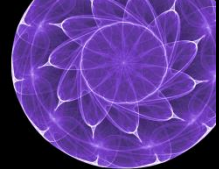
It requires the programmer to define search strategies, thereby losing the declarative nature of CLP.



# Example: N-Queens with CWA

Can we write a set of rules that inductively generates and tests board configurations until it places  $n$  queens so they do not attack?





# Example: N-Queens with CWA

fiveQueens(board)  $\leftarrow$   
safe(board), perm (board, [1,2,3,4,5]).

safe([]).

safe([ $x|y$ ])  $\leftarrow$  noattack( $x, 1, y$ ), safe( $y$ ).

noattack( $x, n, []$ ).

noattack( $x, n, [y|z]$ )  $\leftarrow$

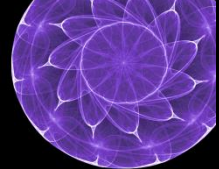
$y \neq x + n, y \neq x - n, \text{noattack}(x, n + 1, z)$ .

perm([], []).

perm([ $x | y$ ],  $w$ )  $\leftarrow$  perm( $y, z$ ), insert ( $x, z, w$ ).

insert( $x, y, [x|y]$ ).

insert( $x, [y | z], [y | w]$ )  $\leftarrow$  insert ( $x, z, w$ ).



# Closed World Query Operation

Use the query operation to search for board configurations

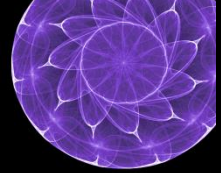
$$query_{cwa}(\Pi, g[\vec{x}]) \doteq \{g[\vec{x} \setminus \vec{t}] \mid g[\vec{x} \setminus \vec{t}] \in lm(\Pi)\}$$

$\Pi$  – The program

$g[\vec{x}]$  – The goal term with variables  $\vec{x}$

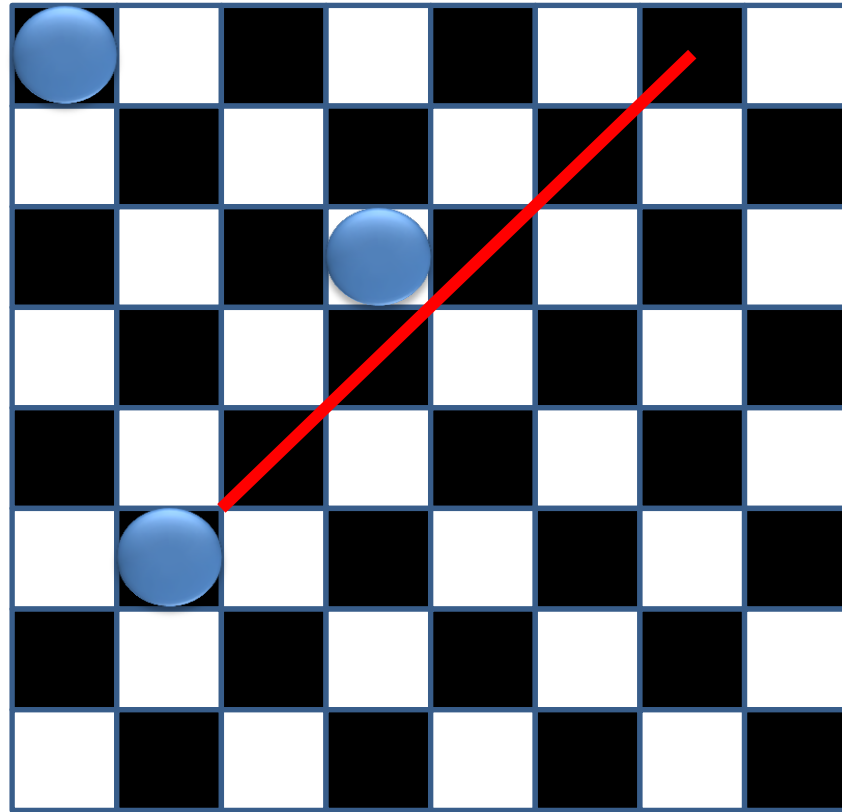
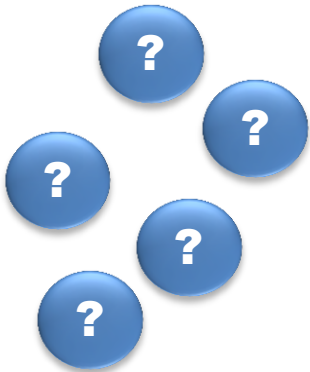
$lm(\Pi)$  – The least model of  $\Pi$

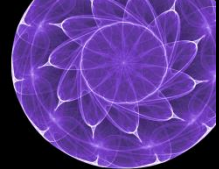
$$query_{cwa}(\Pi_{queens}, fiveQueens(x))$$



# Example: N-Queens with OWA

Give semantics of “attack”. Is there a world containing N queens where none can attack?





# Example: N-Queens with OWA

Size  $\equiv$  size(value: Nat).

Queen  $\equiv$  queen(x: PosInt, y: PosInt).

validSize  $\leftarrow$  size( $m$ ),  $m > 0$ .

validNumQueens  $\leftarrow$  size( $m$ ), count(queen(–,–)) =  $m$ .

bad  $\leftarrow$  size( $m$ ), queen( $x$ , –),  $x > m$ .

bad  $\leftarrow$  size( $m$ ), queen(–,  $y$ ),  $y > m$ .

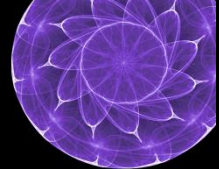
bad  $\leftarrow$  queen( $x$ ,  $y$ ), queen( $x$ ,  $u$ ),  $y \neq u$ .

bad  $\leftarrow$  queen( $x$ ,  $y$ ), queen( $z$ ,  $y$ ),  $x \neq z$ .

bad  $\leftarrow$  queen( $x$ ,  $y$ ), queen( $z$ ,  $u$ ),  $x \neq z$ ,  $x - y = z - u$ .

bad  $\leftarrow$  queen( $x$ ,  $y$ ), queen( $z$ ,  $u$ ),  $x \neq z$ ,  $x + y = z + u$ .

fiveQueens  $\leftarrow$  size(5), validSize, validNumQueens, **fail** bad.



# Open World Query Operation

Use the query operation to search for board configurations

$$query_{owa}(\Pi, g[\vec{x}]) \doteq \{(g[\vec{x} \setminus \vec{t}], W) \mid g[\vec{x} \setminus \vec{t}] \in lm(\Pi \cup W)\}$$

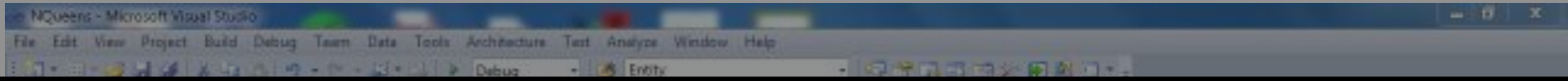
$W$  – A finite extension  $\Pi$  of by facts.

May further constraint the query operation so every  $W$  is a subset of regular Herbrand set.

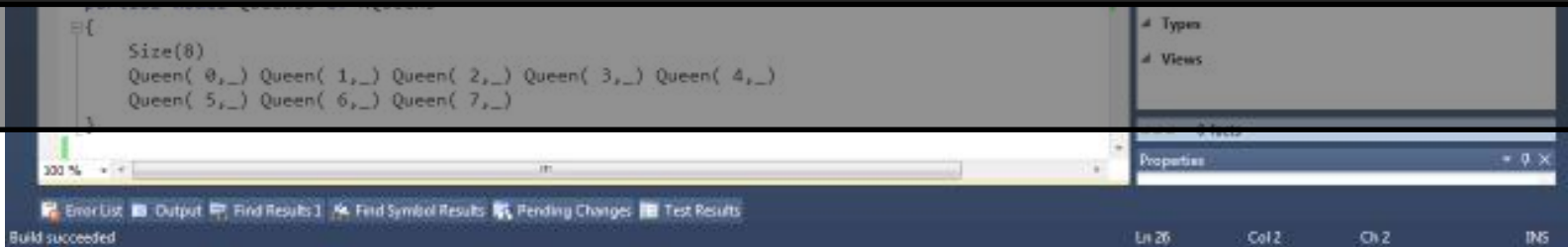
$$\begin{aligned} & query_{owa}(\Pi_{queens}, fiveQueens) \\ & \quad \text{such that} \\ & \quad W \subseteq \llbracket Queen \cup Size \rrbracket \end{aligned}$$



# Finding Models



```
model Queens8_1 of NQueens at "../NQueens.4ml"  
{  
  Size(8)  
  Queen(0,4)  Queen(1,0)  Queen(2,3)  Queen(3,5)  
  Queen(4,7)  Queen(5,1)  Queen(6,6)  Queen(7,2)  
}
```



<http://research.microsoft.com/formula>

## iii. Design and Implementation

<http://research.microsoft.com/formula>

**FORMULA**

Modeling Foundations.





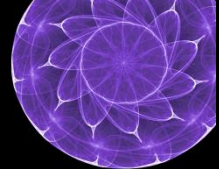
# Types in OWA

Need semantic types to explain the shape of worlds, otherwise we search many erroneous words:

*Size(Queen(bad, false))*

Also want to know if the OWA program is meaningful over all well-typed worlds.  
Must deal with constraints.

Use a sub-class of regular types admitting canonical forms.  
Lift interpreted functions into the type system via Galois approximations.



# Type Language

Type terms  $\tau ::= \alpha \mid \beta \mid c \mid \tau \cup \tau \mid \tau \cap \tau \mid f(\tau^+) \mid o(\tau^+)$

Type Decls  $D ::= \alpha \equiv \tau$

Given the declarations:

$$\alpha_{cons} \equiv cons(Integer, \alpha_{list}). \quad \alpha_{list} \equiv nil \cup \alpha_{cons}.$$

Then this is a type term:

$$\alpha_{list} \cap \left[ \begin{array}{l} cons(0, \alpha_{list}) \cup cons(Integer, nil) \cup \\ cons(PosInteger \cup NegInteger \cup String, cons(Integer, \alpha_{list})) \end{array} \right]$$

And its canonical form is:

$$\alpha_{cons}$$



# Constraint-Level Type Inference

Need to determine if a rule is meaningful over all possible worlds.

$$F \equiv f(\text{arg}: \text{Real} \cup F)$$

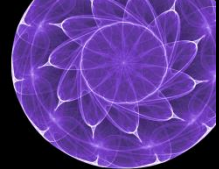
$$f(x.\text{arg}) \leftarrow f(x), f(y), x \in \text{Nat} \cup f(\text{Nat}), y \in \text{Nat} \cup f(\text{Nat}), \\ x.\text{arg} = y.\text{arg}.\text{arg} + 1.$$

Over-approximate the satisfying assignments of the body with regular types

$$x : f(\text{PosInt}), \quad y : f(f(\text{Nat})).$$

Check that the head is a sub-type of the declared type.

$$f(\pi_1(x))[x \setminus f(\text{PosInt})] = f(\pi_1(f(\text{PosInt}))) = f(\text{PosInt})$$



# Model Finding By Symbolic Execution

Guess a symbolic world and symbolically execute logic program:

$$W_5 \doteq \left\{ \begin{array}{l} \text{size}(5), \\ \text{queen}(1, c_1), \text{queen}(2, c_2), \\ \text{queen}(3, c_3), \text{queen}(4, c_4), \text{queen}(5, c_5) \end{array} \right\}$$

The state of symbolic execution is represented by:

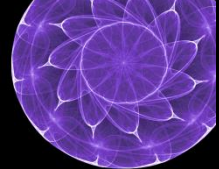
$$\mathbf{H} \doteq \{(t_1, \rho_1), (t_2, \rho_2), \dots\},$$

Each  $t_i$  and  $\rho_i$  are terms and quantifier-free formulas over symbolic values.

$$\mathbf{H}_{ex} \doteq \left\{ \begin{array}{l} (\text{queen}(1, c_1), \text{true}), (\text{queen}(2, c_2), \text{true}), \\ (\text{bad}, c_1 = c_2) \end{array} \right\}.$$

$$\textit{knows}(\mathbf{H}_{ex}, (1, 2)) = \{\text{queen}(1, 1), \text{queen}(2, 2)\},$$

$$\textit{knows}(\mathbf{H}_{ex}, (1, 1)) = \{\text{queen}(1, 1), \text{queen}(2, 1), \text{bad}\}.$$



# Model Finding By Symbolic Execution

$query_{owa}(\Pi_{queens}, fiveQueens)$  such that  $W \subseteq \llbracket Queen \cup Size \rrbracket$

$(fiveQueens, \rho_{5queens}) \in H^\infty$

Boolean connectives

Type coercions

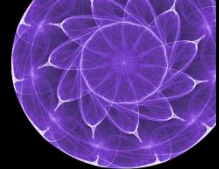
Symbolic constants

Arithmetic functions

```

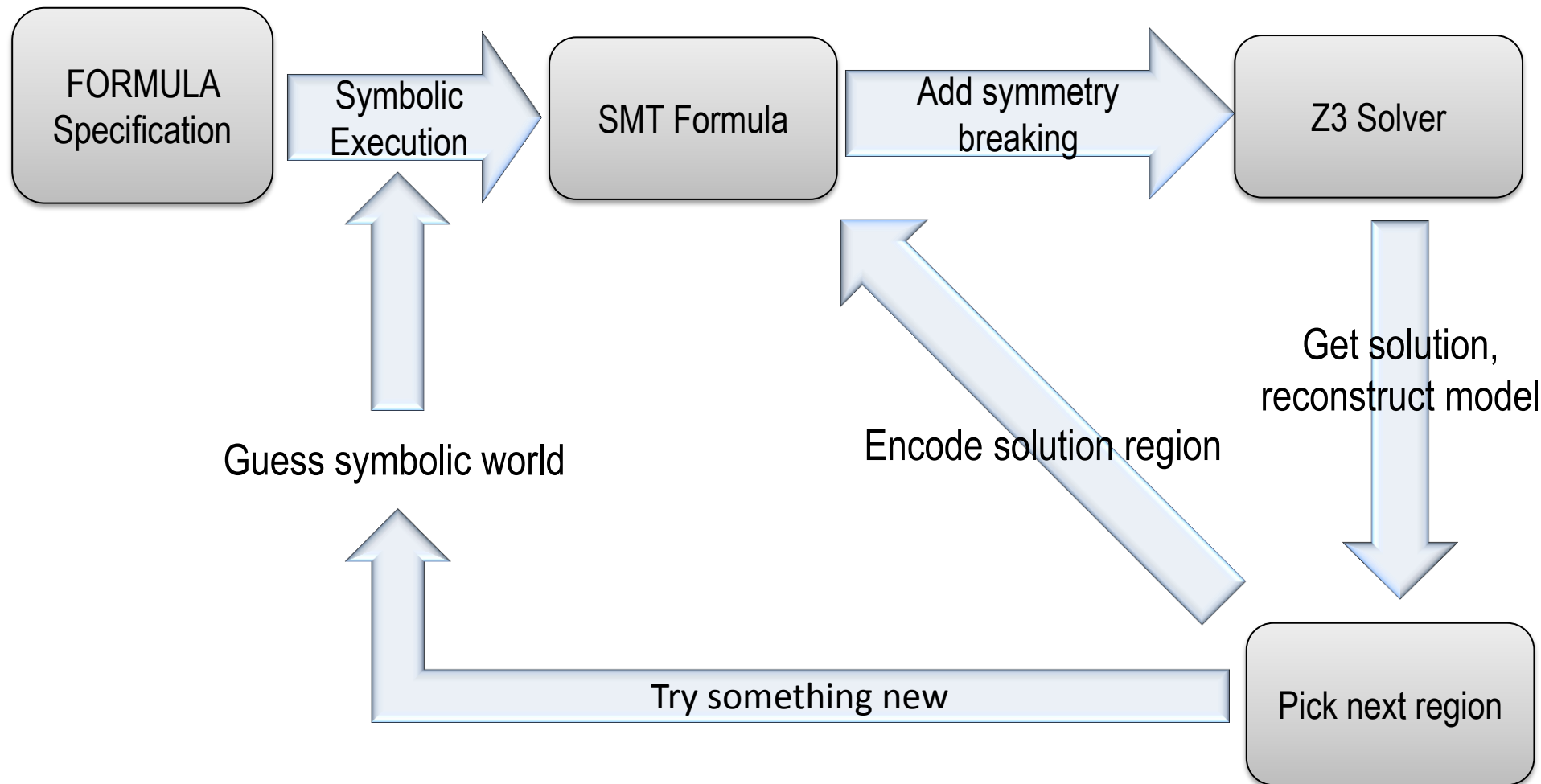
(not (or (>= (@Nat2Int _<42>!22) 5) (= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<46>!25)))) -2)
(>= (@Nat2Int _<43>!23) 5) (= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<47>!26)))) -3)
(>= (@Nat2Int _<45>!24) 5) (= (+ (@Nat2Int _<45>!24) (* -1 (@Nat2Int _<46>!25)))) -1)
(>= (@Nat2Int _<46>!25) 5) (= (+ (@Nat2Int _<45>!24) (* -1 (@Nat2Int _<47>!26)))) -2)
(>= (@Nat2Int _<47>!26) 5) (= (+ (@Nat2Int _<46>!25) (* -1 (@Nat2Int _<47>!26)))) -1)
(= _<42>!22 _<43>!23) (= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<43>!23)))) 1)
(= _<42>!22 _<45>!24) (= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<45>!24)))) 2)
(= _<42>!22 _<46>!25) (= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<46>!25)))) 3)
(= _<42>!22 _<47>!26) (= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<47>!26)))) 4)
(= _<43>!23 _<45>!24) (= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<45>!24)))) 1)
(= _<43>!23 _<46>!25) (= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<46>!25)))) 2)
(= _<43>!23 _<47>!26) (= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<47>!26)))) 3)
(= _<45>!24 _<46>!25) (= (+ (@Nat2Int _<45>!24) (* -1 (@Nat2Int _<46>!25)))) 1)
(= _<45>!24 _<47>!26) (= (+ (@Nat2Int _<45>!24) (* -1 (@Nat2Int _<47>!26)))) 2)
(= _<46>!25 _<47>!26) (= (+ (@Nat2Int _<46>!25) (* -1 (@Nat2Int _<47>!26)))) 1))
(= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<43>!23)))) -1)
(= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<45>!24)))) -2)
(= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<46>!25)))) -3)
(= (+ (@Nat2Int _<42>!22) (* -1 (@Nat2Int _<47>!26)))) -4)
(= (+ (@Nat2Int _<43>!23) (* -1 (@Nat2Int _<45>!24)))) -1)

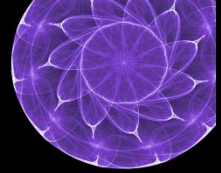
```



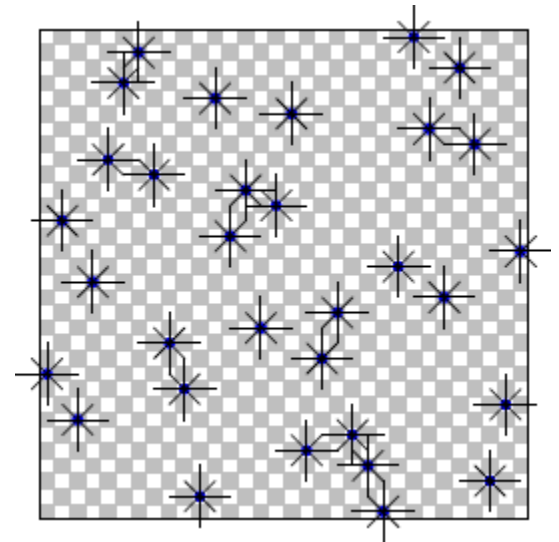
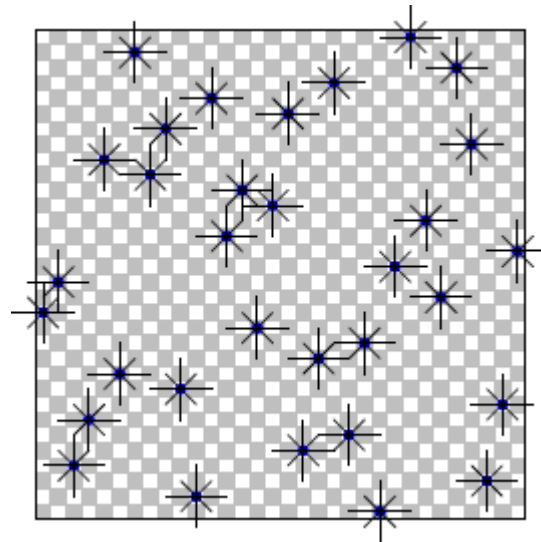
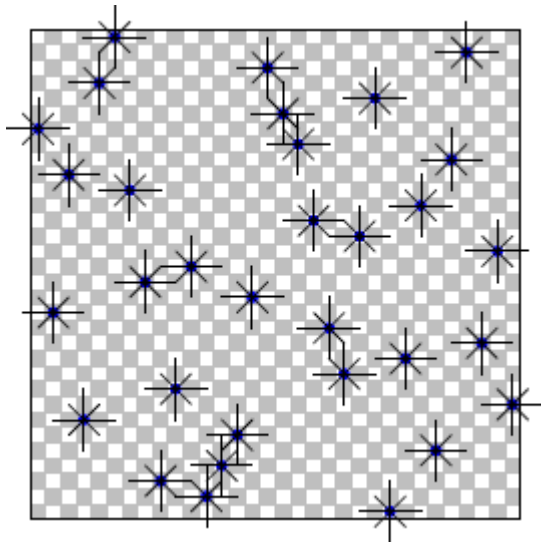
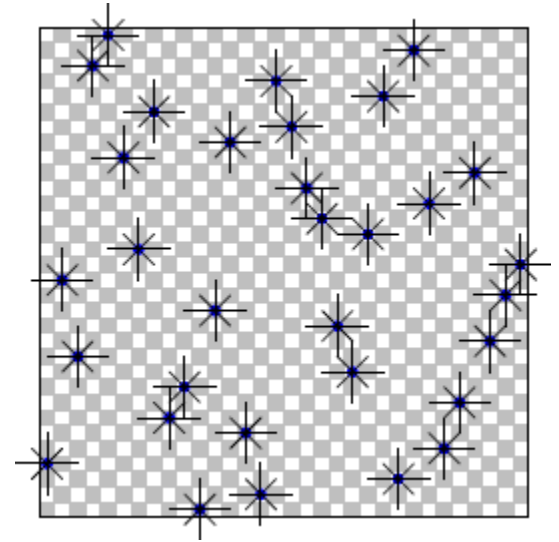
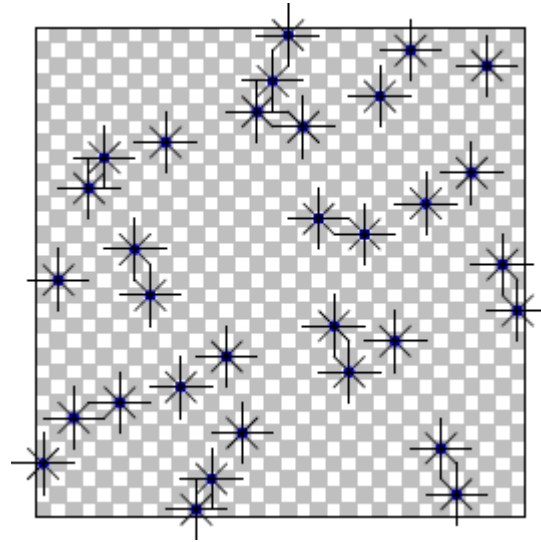
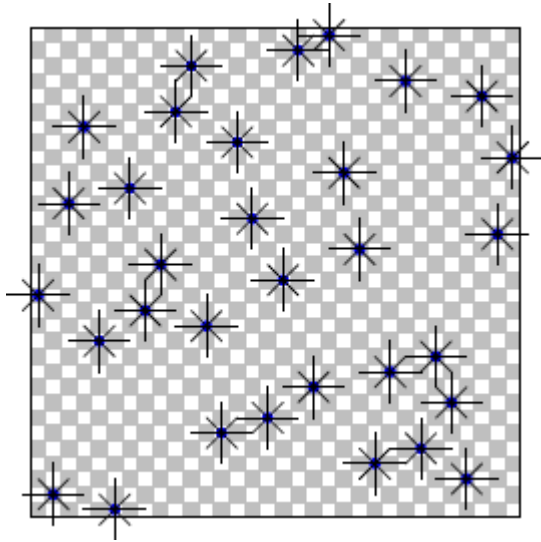
# Solving and Search

Use state-of-the-art **satisfiability modulo theories (SMT)** solver Z3 to solve quantifier-free formulas.





# Solving and Search



Questions?

<http://research.microsoft.com/formula>

**FORMULA**

Modeling Foundations.

