



How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

How Formal Methods Save Christmas



A Christmas Story
by Jan O. Ringert





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Santa Claus in Despair

It was a cold and icy winter morning a few days before Christmas. The land was still dark and almost everybody was asleep. However, far north on the North Pole a great hustle and bustle was going on. Santa Claus, his elves and fairies were preparing for Christmas and this year it seemed to be even busier than all the years before.

Aside from all the elves running around, Santa Claus and two of his oldest friends – the elves Zip and Zap – were standing together in a relatively quiet corner. Santa had a worried look on his face.



Santa: “This year things are really getting out of hand. It is only a few more days until Christmas and we are by far not as prepared as we should be.”

Zip: “We were kind of taken by surprise by the enormous amount of wish lists and requests we got from all over the world.”

Zap: “There is a general problem with this. When we started business around the 15th century, there was only half a billion people on earth. This year we hit the seven-billion-people barrier and they keep multiplying.”





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Zip: “It is an exponential growth and additionally more and more countries are celebrating Christmas. We are experiencing extreme delays in processing all the wish lists. There are frequent jams when transporting goods to the present factories on the polar circle and back just because of so much reindeer sleds traffic.”

Santa: “I am afraid we have to call Christmas off this year... I do not see any way we can finish our work in time. Maybe Christmas can only be celebrated every second year, now...”

Zap: “Wait, I have an idea. Humans also have problems with growing complexities. I once met a person who works at a company that develops software. They have similar problems. Their systems get so big, complex and safety critical that they can no longer maintain them with the methods they used a decade ago. I think they got help from the Formal Methods.”

Zip: “Let’s invite the Formal Methods to the North Pole and see how they can help us!”

Santa agreed that they needed help. Maybe getting help from Formal Methods was the last possible way that he and his team could finish in time. He invited them over and not much later Alloy, FORMULA, SAT solver, and SMV arrived at the North Pole.





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Formal Methods Character Introduction

This is a rather short and shallow overview of the Formal Methods characters in this Christmas story. The selection is based on subjective experience and does not adhere to any criteria. The presentation even more so...



Alloy is the name of a specification language used for modeling systems based on relational first order logic. The building blocks of modeled systems are signatures with inheritance and fields that give Alloy an object-oriented flavor. Facts and predicates state desired or unwanted properties of instantiations of the signatures and their relations. In a bounded scope (fixed upper limit to the number of atoms of each signature), the Alloy language is decidable and properties can be checked automatically using the Alloy Analyzer.

Alloy proves wrong an old and constant misconception: computer science people are (!) capable of handling relations rather well!

SMV stands for symbolic model verifier and is a tool for symbolic model checking of state transition systems (STS). States are assignments to variables and a transition relation defines how to change these values in transition steps. SMV checks temporal logic formulas (written in CTL or LTL) formulas, which are assertions on properties of infinite executions of the STS.



SAT solvers compute assignments of variables satisfying propositional formulas. Computing these assignments is NP-hard in the general case but of great relevance in applications, since many other problems can be reduced to SAT problems. SAT solver developers have optimized solving these hard problems and thus provide strong off-the-shelf “workhorses”.

In the end it is all about satisfaction.

FORMULA is the “youngest” formal methods tool in this short list. A problem domain is defined as a set of algebraic data types. FORMULA computes well-formed models using techniques from constraint logic programming. FORMULA supports the automated generation of valid models and the completion of partial models as well as the formalization of model transformations.

FORMULA is a model finder spending his days looking for beautiful well-formed models (not quite like Heidi Klum or Tyra Banks, but similar – he just searches in other domains).





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Processing Wish Lists (SAT solver)

When arriving at the North Pole the Formal Methods meet Mrs. Santa in the post office of the North Pole. Mrs. Santa reads every wish list that arrives and decides which presents need to be wrapped and shipped. Over the past years, Santa Claus and his elves had stocked more and more different kinds of presents and children's wishes had become more and more complicated.

Mrs. Santa: "I am opening all these letters but instead of wishing for a simple thing like a puppy or a doll these wish list letters get more and more confusing because of the many options children have. Here is an example of what a child wrote."

Dear Santa Claus,

I wish for a puppy or a doll, and a PS3 or an Xbox. If you get me a puppy, I also want a dog basket and a ball. If I get the doll, I want dresses for the doll and the PS3 Barbie game. If I get the puppy I don't want the PS3 since only the Xbox has the dog trainer game I want. Only if I get the doll I want a dollhouse.

Thank you,
Julie

Mrs. Santa: "... and this is one of the simpler ones. I have to figure out exactly what presents to give to these kids to not disappoint them. Can you help me out?"

SAT solver: "No problem, I do things like that all day. Let me show you how..."

```
(puppy ∨ doll) ∧ (PS3 ∨ Xbox)
puppy ⇒ dogBasket ∧ ball
doll ⇒ dresses ∧ barbieGame
puppy ⇒ ¬PS3 ∧ Xbox ∧ dogTrainerGame
doll ⇔ dollHouse
```

SAT solver: "First we write down the text as propositional formulas where all the items mentioned in the list are truth variables. If the variable of an item is true the child will get it for Christmas, if the variable's value is false it will not receive this item as a gift."

SAT solver: "Now we express these formulas in the conjunctive normal form (CNF) which is a conjunction of disjunctions of variables and negated variables."





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Mrs. Santa: "I remember the conversion to CNF from my evening classes. An implication $puppy \Rightarrow dogBasket$ is translated to $\neg puppy \vee dogBasket$ and with the use of this other rule $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$ we can easily convert the wish list." (See the left formula.)

$puppy \vee doll$	\wedge	1 2 0
$PS3 \vee Xbox$	\wedge	3 4 0
$\neg puppy \vee dogBasket$	\wedge	-1 5 0
$\neg puppy \vee ball$	\wedge	-1 6 0
$\neg doll \vee dresses$	\wedge	-2 7 0
$\neg doll \vee barbieGame$	\wedge	-2 8 0
$\neg puppy \vee \neg PS3$	\wedge	-1 -3 0
$\neg puppy \vee Xbox$	\wedge	-1 4 0
$\neg puppy \vee dogTrainerGame$	\wedge	-1 9 0
$\neg doll \vee dollHouse$	\wedge	-2 10 0
$\neg dollHouse \vee doll$		-10 2 0

SAT solver: "Yes, it is as easy as that. To the right of your formula you can see the DIMACS input format that I normally use. Variables are replaced by numbers, negation by the minus sign, every line is a disjunction, and all lines are conjuncted. Here is a solution: the child should get a doll, a PS3, the dresses, the Barbie game and the doll house."





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Organizing Secret Santa (Alloy)

After having heard about the initial success of Formal Methods, Santa Claus invited the Formal Methods to his office where even more mail had piled up.

Santa: “We are responsible for all kinds of things around Christmas. There are many traditions – some of them complicated – and if people need help or advice, they send me letters. There are numerous requests of offices, clubs and companies that want to organize a Secret Santa Game.”

SMV: “But that sounds really nice. This saves you some work if people give presents to each other...”

Santa: “It sure is nice. The problem however is that they sometimes have trouble finding out who should buy a present to whom. They ask me for assigning people and many times it really gets complicated like in this request.”

Dear Santa Claus,

We want to organize a Secret Santa in our office. However, some restrictions make it hard for us to find assignments of who prepares presents for whom:

1. No intern should have to buy a present for her/his supervisor.
2. The IT department employees should not be Secret Santa for female employees because they never know what to buy for women.
3. Managers should get presents from the secretaries since they know best.
4. Nobody should get a person assigned that she or he shares an office with.

Can you please help us with this task?

Find the employee list attached.

Sincerely,

Max

(in charge of the companies Secret Santa)

Name	Position	Department	Office partners
Abigail	Manager	IT	None
Jon	Manager	Accounting	None
Andrew	Secretary	IT	Anna
Anna	Secretary	Accounting	Andrew
Max	Intern	IT	Jane (supervisor), Otto
Jane	Programmer	IT	Max, Otto
Otto	Writer	Marketing	Max, Jane
Tom	Writer	Marketing	None
...





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Alloy: “Consider this problem to be solved. I can show you how to model this and probably give you a number of solutions to this problem.”



Alloy: “I will first model the employees that we are dealing with as signature `Employee` with a `gender`, a `department`, their office partners, and the person they are the Secret Santa of. `Gender` and `Department` are simple enumerations of values that these fields can have. Secretaries, managers, programmers, writers and interns are special employees. Interns have an extra field pointing to their supervisor.”

```
abstract sig Employee {
  gender : one Gender,
  department : one Department,
  officePartners : set Employee,
  isSecretSantaOf : one Employee
}

abstract sig Secretary, Manager, Programmer,
  Writer extends Employee {}

abstract sig Intern extends Employee {
  supervisor : one Employee
}

enum Gender {male, female}
enum Department {Accounting, IT, Marketing, HR}
```

Santa: “What about the obvious rules of the Secret Santa game, like nobody is her/his own Secret Santa and everybody has exactly one Secret Santa?”





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Alloy: “We already made sure that every employee has exactly one Secret Santa by using the multiplicity **one** above. The other fact is shown below: there is no `Employee` which is his own Secret Santa.”

```
fact SecretSantaNotSelf {
    no e : Employee |
        e.isSecretSantaOf = e
}
```

Santa: “So far so good, now we need the special conditions of this company’s Secret Santa. I suppose we do it in a similar way?”


Alloy: “Absolutely, I will also show you the facts expressing these rules:”

```
fact NoInternSecretSantaOfDirectSupervisor {
    no i : Intern |
        i.isSecretSantaOf = i.supervisor
}

fact NoITEmployeeIsSecretSantaOfFemale {
    no e : Employee |
        e.department = IT and
        e.isSecretSantaOf.gender = female
}

fact AllManagersGetPresentsFromSecretaries {
    Manager in Secretary.isSecretSantaOf
}

fact NoSecretSantaInSameOffice {
    all e : Employee |
        e not in e.isSecretSantaOf.officePartners
}
```

 Santa: “Impressive... I like the notation in the third fact saying that the set of all `Managers` is a subset of the `Employees` that the `Secretaries` are Secret Santa of. But how do we encode the list of employees that we received with the letter?”

Alloy: “That is also quite simple. Look at the following example.”



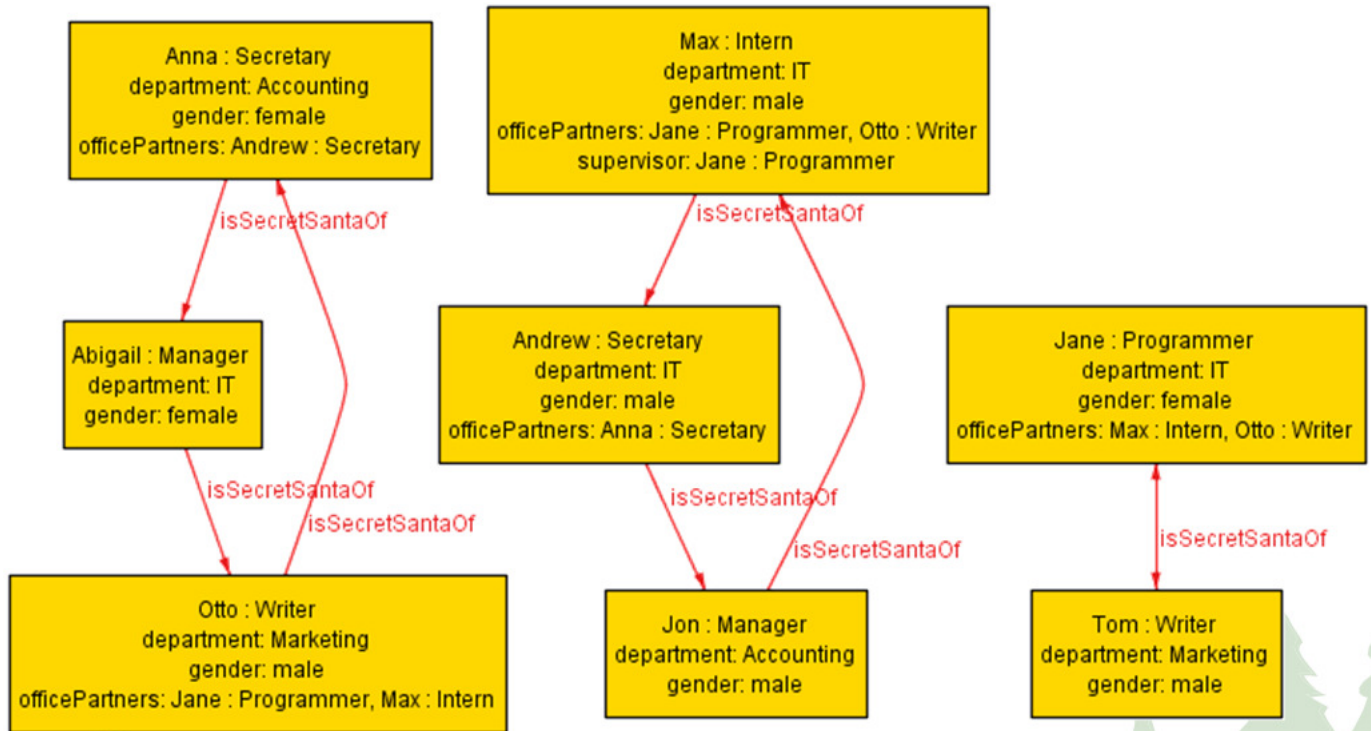


How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

```
one sig Jon extends Manager {} {  
  gender = male  
  department = Accounting  
  no officePartners  
}  
one sig Anna extends Secretary {} {  
  gender = female  
  department = Accounting  
  officePartners = Andrew  
}
```

Alloy: “Now give me a few milliseconds and I can give you an answer to the question who should be the Secret Santas.”



Santa: “Thank you so much. You really helped us out. Trying to figure out these things by hand is not fun.”





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Guiding the Sleds (SMV)

While Santa and Alloy worked on matching the Secret Santas all over the world, SMV and FORMULA met Zip and Zap outside where the two elves presented them the currently most pressing problem concerning the logistics on the North Pole. Many of the toy factories were situated near the polar circle around the North Pole. All the traffic from and to these places had to come across the central junction of the Christmas village.

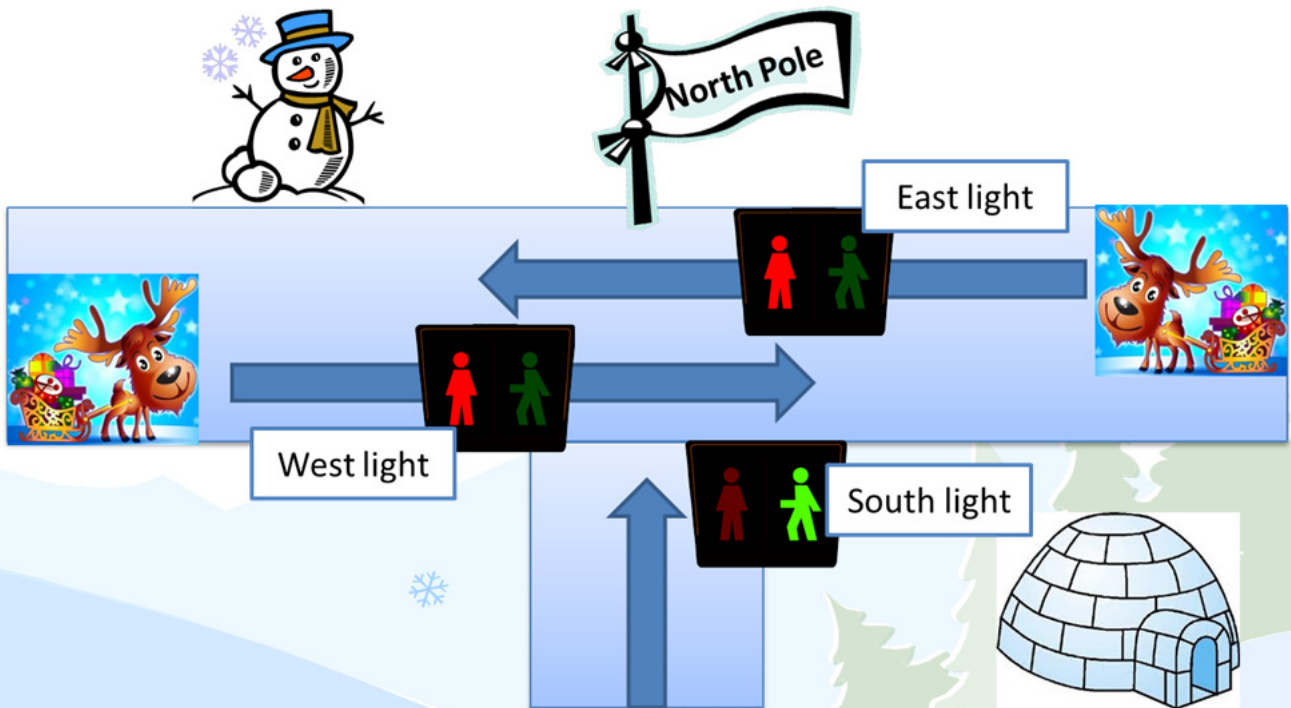
Zip: “There is so much more traffic than there used to be. All roads are crowded and sleds often crash or stop all at the same time to wait for the others to pass.”

Zap: “We are thinking about a new traffic guidance system. We have already installed lights and traffic sensors but we don’t know yet how to control them in order to make the traffic flow lively and safely.”

FORMULA: “Well then, show us the junction, maybe we can help.”

Zip: “There it is. It is a three-way junction with a lane going from east to west, one from west to east, and a third one coming from the south. The lanes to the west and east can be used simultaneously but not if there is traffic coming from the south.”

Zap: “A few years ago we didn’t need these stupid traffic lights; there was few enough traffic for the three lanes to handle without the need of traffic lights.”



FORMULA: “That is only a three way crossing why don’t you add a fourth direction maybe this solves the problem by increasing the throughput?”





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Zap: “Well, maybe I can give you an advice since you are new here: *Never try to go north on the North Pole!*”

SMV: “Calm down, I can help you with this. Let us start by designing a control system for the traffic lights. Later we can formulate your requirements about its correctness that I will then help you prove or invalidate these.”

```
VAR
WestLightGreen      : boolean;
EastLightGreen      : boolean;
SouthLightGreen     : boolean;
ReindeerWest        : boolean;
ReindeerEast        : boolean;
ReindeerSouth       : boolean;
EWLock              : boolean;

INIT
!WestLightGreen & !EastLightGreen & !SouthLightGreen & !EWLock;

DEFINE
WestCanGo := ReindeerWest & !WestLightGreen & !ReindeerSouth;
WestDone  := !ReindeerWest & WestLightGreen;
EastCanGo := ReindeerEast & !EastLightGreen & !ReindeerSouth;
EastDone  := !ReindeerEast & EastLightGreen;
```

SMV: “Here is the basic setup. The traffic lights on the intersection allow reindeer to pass when we set the corresponding variable WestLightGreen, EastLightGreen or SouthLightGreen. We have sensors that notify us about reindeer waiting: ReindeerWest, ReindeerEast and ReindeerSouth. The variable EWLock should be when traffic flows from the east west direction to prevent traffic from the south.”

Zip: “I see that initially the crossing does not allow any traffic.”

Zap: “And there are some definitions about when a reindeer can go west (WestCanGo): there needs to be a reindeer waiting, the light must be red and there should not be a reindeer waiting in the south.”

SMV: “That is correct. Now we define how the variables can change from one state to the next. Here is an obvious fact: if a reindeer waits at a light and the light is not green, the reindeer will still be there in the next step.”

```
TRANS
(ReindeerWest & !WestLightGreen -> next(ReindeerWest)) &
(ReindeerEast & !EastLightGreen -> next(ReindeerEast)) &
(ReindeerSouth & !SouthLightGreen -> next(ReindeerSouth));
```





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Zap: “I now got the idea with the lock. When the traffic from the east can go, we allow it to go but also set the lock. If all the traffic from the east is done we disallow the traffic and reset the lock.”

Zap: “Wait, let us keep the lock if the opposite light on the western side still allows reindeer to pass.”

Zap: “Sure, the control for the light in the west is then analogous and for the light in the south we can allow traffic if the crossing is not locked.”

TRANS

case

```
EastCanGo: next(EastLightGreen) & next(EWLock);
EastDone: !next(EastLightGreen) &
  (next(EWLock) = WestLightGreen);
TRUE: (next(EastLightGreen) = EastLightGreen);
esac;
```

...

TRANS

case

```
ReindeerSouth & !EWLock & !SouthLightGreen: next(SouthLightGreen);
SouthLightGreen & !ReindeerSouth: !next(SouthLightGreen);
TRUE : next(SouthLightGreen) = SouthLightGreen;
esac;
```

SMV: “Very good, let us verify this system. We want to check that no reindeer sleds crash: no states should allow traffic from the south together with traffic from the west or east. Furthermore, no reindeer should wait forever. We assume that reindeer waiting in a direction with a green light means that eventually all reindeer have passed. Under this assumption whenever a reindeer waits at the traffic lights, it will eventually be allowed to pass.”

```
LTLSPEC NAME noReindeerCrash :=
  G !(SouthLightGreen & (WestLightGreen | EastLightGreen));

LTLSPEC NAME noReindeerWaitsForever :=
  G (ReindeerWest & WestLightGreen -> F !ReindeerWest) &
  G (ReindeerEast & EastLightGreen -> F !ReindeerEast) &
  G (ReindeerSouth & SouthLightGreen -> F !ReindeerSouth) ->

  G (ReindeerWest -> F WestLightGreen) &
  G (ReindeerEast -> F EastLightGreen) &
  G (ReindeerSouth -> F SouthLightGreen);
```

Zip: “Is our work correct?”





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

SMV: “I am afraid it is not. When the two traffic lights on the east and the west side both turn to red simultaneously, they do not remove the lock. This blocks the southern reindeer. Their request to pass in turn blocks the others which leads to a deadlock.”

Zip: “Can we help these poor guys?”



SMV: “Sure we can, we just have to change the condition for removing the lock. We should only keep the lock when the opposite direction is allowed to pass and the reindeer are not done crossing: `next(EWLock) = WestLightGreen & !WestDone`. It is analogous on the eastern side.”

[This problem is based on the traffic lights example from the tutorial K.L. McMillan, *Getting started with SMV*, <http://kenmcmil.com/tutorial.ps>, 1999.]





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

Setting up Christmas Trees (FORMULA)

While SMV helped Zip and Zap to control the safe flow of reindeer sleds traffic, FORMULA wandered on to look for more elves that were in need of the Formal Methods' help. He found a room full of maps and large clouds that came from smoking elves' heads. He entered the room and one of the elves looked up from his maps and tons of disposed scratch paper.

Tifill: "Hello FORMULA, we hear that you guys are good at solving complex problems. We got one right here."

FORMULA: "What is your problem? I will see how I can help."

Tifill: "I am Tilfill and I am responsible for helping cities and big companies in planning their Christmas decoration. There is this big city that has a square marketplace which is 8 by 8 units..."

FORMULA: "Sounds like a big chess board..."

Tifill: "They want to set up 8 Christmas trees but all of them should be visible from all rows, columns, and diagonals – no two trees should cover each other."

FORMULA: "I am quite confident that I can help you with this task. Let me show you how to express these conditions."





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

FORMULA: “The domain that we are talking about is the domain of Christmas trees. A Christmas tree has an x- and a y-coordinate marking the position on the market place where the tree is placed.”

Tifill: “There are some undesired placements of trees we should detect: a tree is placed outside the market place if the x-coordinate is greater or equal to the number of trees (coordinates start at 0). The same is true if there is a tree with a y-coordinate greater or equal to the number of trees.”

```
domain ChristmasTrees
{
  primitive ChristmasTree ::= (x: Natural, y: Natural).

  OutsideMarketPlace := ChristmasTree (x,_),
                        x >= count(ChristmasTree(_,_)) .
  OutsideMarketPlace := ChristmasTree (_,y),
                        y >= count(ChristmasTree(_,_)) .

  SameRow      := t1 is ChristmasTree, t2 is ChristmasTree,
                  t1.y = t2.y, t1 != t2.
  SameColumn   := t1 is ChristmasTree, t2 is ChristmasTree,
                  t1.x = t2.x, t1 != t2.
  SameDiagonal := t1 is ChristmasTree, t2 is ChristmasTree,
                  t1 != t2, t1.x - t1.y = t2.x - t2.y.
  SameDiagonal := t1 is ChristmasTree, t2 is ChristmasTree,
                  t1 != t2, t1.x + t1.y = t2.x + t2.y.

  conforms := !OutsideMarketPlace &
              !SameRow & !SameColumn & !SameDiagonal.
}

[Cardinality(ChristmasTree, 8)]
partial model TreeDistribution of ChristmasTrees {}
```

FORMULA: “To prevent that one of the 8 trees is hidden behind another one we don’t want two trees to be in the same row, column or diagonal. I wrote this down as deduction rules where the left side (e.g., SameRow or SameDiagonal) is established to be true only if it has a right side that can be matched to appropriate parts of the model of the market place.”

Tifill: “This means that SameRow would be true if there would be two distinct Christmas trees ($t1 \neq t2$) that have the same y-coordinate?”

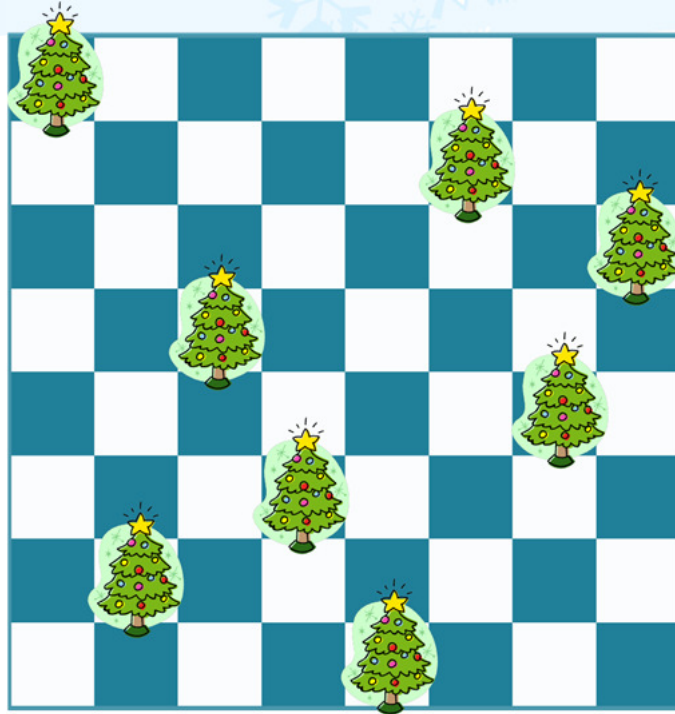




How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

FORMULA: “Yes, this is what it means. Now we want to find a model of the market place that conforms to our domain where none of these unwanted properties are true. Here is one (out of 92 different placements of trees):”



[This problem is based on the N-Queens example by the FORMULA project, <http://research.microsoft.com/en-us/um/redmond/projects/formula/>]





How Formal Methods Save Christmas

<http://christmas.formal-methods.net>

There were still many things to do on the North Pole to prepare for Christmas but now Santa Claus was sure they would be able to make it in time. This way Formal Methods saved Christmas and proved that they cannot only put a smile on theoreticians' faces, but also those of children.



The End

P.S.: You can find the complete source code of all the examples and some helpful links to tools on the web (<http://christmas.formal-methods.net>).

