Symmetry in Model Checking

Matthew Gwynne

Model Checking Workshop

2nd December 2010
The main problem in model checking that one keeps coming back to,
The main problem in model checking that one keeps coming back to, over
The main problem in model checking that one keeps coming back to, over and over
The main problem in model checking that one keeps coming back to, over and over ... and over again is
The main problem in model checking that one keeps coming back to, over and over ... and over again is

**The State Space Explosion!**
We’ve seen numerous methods for dealing with such complexities, including -

- Symbolic model checking
- Partial order reduction
- Compositional reasoning
- Abstraction
- ... etc!
However,
However, imagine we have

1. a set of lights,
2. these lights begin in the off state,
3. the lights can be switched from off to on,
4. only one light can be turned on at a time,
5. and eventually we want all lights to be on.
Motivation
What is symmetry?
Applications
Motivation
What is symmetry?
Applications

All on! :)}
Motivation

What is symmetry?

Applications

All on! :)

[Image of two light bulbs, indicating symmetry]
All on! :)
If we model this, we get a state space something like this -

![Transition graph for light switching](image)

**Figure**: Transition graph for light switching
But we don’t need that much! Order doesn’t matter!

Any light is the same as any other!
But we don’t need that much! Order doesn’t matter!

Any light is the same as any other!

We only need to know the **number** of lights!
Figure: Transition graph for light switching

Figure: “Quotiented” Transition graph for light switching
The point is, there is a **symmetry** here:

**Figure:** Transition graph for light switching (1)  
**Figure:** Transition graph for light switching (2)  
**Figure:** Transition graph for light switching (3)
I can turn them on, one way
Motivation
What is symmetry?
Applications

I can turn them on, one way
I can turn them on, one way
I can turn them on, one way
I can turn them on, one way

All on! :}
or another
or another
or another
or another
or another

All on! :)

Motivation
What is symmetry?
Applications
Motivation
What is symmetry?
Applications

It’s all the same!
Motivation
What is symmetry?
Applications

It’s all the same!

However you swap things around!
It’s all the same!

However you swap things around!

Everything is symmetric
Motivation

What is symmetry?

Applications
What is symmetry?
Motivation
What is symmetry?
Applications

Intuition for symmetry
Same here:

**Figure:** Transition graph for light switching (1)

**Figure:** Transition graph for light switching (2)

**Figure:** Transition graph for light switching (3)
The point is in each case we have *mappings* from the object to itself which *preserves* the structure.
The point is in each case we have *mappings* from the object to itself which *preserves* the structure.

We have **automorphisms** - *functions* from a set of objects to itself which preserves the structure.
An automorphism is a map from an object to itself which preserves the structure.

So for a Kripke structure \((S, R, L)\) we have that \(\pi : S \to S\) is an automorphism if

\[
(s, t \in S \land (s, t) \in R) \Rightarrow (\pi(s), \pi(t)) \in R
\]

That is, \(\pi\) must maintain the transition relation.
There are often many automorphisms (symmetries) of any structure.

Given any two automorphisms $\pi_1$ and $\pi_2$, composing them yields an automorphism.

This leads us to an **automorphism group** for a structure, a set of automorphisms on the structure, along with function composition, which obeys the usual **group** properties (addition, inversion, associativity, identity etc).
Automorphism example

Figure: Transition graph for light switching (1)

Figure: Transition graph for light switching (2)
The orbit of $x \in X$ with respect to the permutation group $G$ on $X$ is the set of elements which map to $x$ given some permutation in $S_X$. So

$$O_G(x) := \{ y \in X \mid \exists \pi \in G : \pi(y) = x \}$$

So we collect all states together which are essentially the same.
Quotient models

The idea here is that we want to represent all “similar” states by just one state, so we replace them by a set of states which then represents the whole set.

Figure: Transition graph for light switching

Figure: “Quotiented” Transition graph for light switching
However, if actually represented each set, the states themselves might be quite large. Therefore we represent the whole set of elements by a single element.

So, we have a function $\text{rep} : \mathcal{O}_G(S) \rightarrow S$ which maps any orbit to a single state which represents that orbit.
However, if actually represented each set, the states themselves might be quite large. Therefore we represent the whole set of elements by a single element.

So, we have a function $\text{rep}: \mathcal{O}_G(S) \rightarrow S$ which maps any orbit to a single state which represents that orbit.

For example, in our lightbulb example, we might map $\{110, 011, 101\}$ to 110 (i.e. we sort it).
Therefore, given a Kripke structure \((S, R, L)\), and an automorphism group \(G\) which acts on \((S, R, L)\), the quotient model (using representatives) for \((S, R, L)\) w.r.t \(G\) is \((S_G, R_G, L_G)\) where

- \(S_G := \{\text{rep}(O_G(x)) \mid x \in S\}\)
- \(R_G := \{(\text{rep}(O_G(x)), \text{rep}(O_G(y))) \mid (x, y) \in R\}\)
- \(L_G(\text{rep}(O(x))) := L(\text{rep}(O(x)))\)
However, something is not quite right...
However, something is not quite right...

If I had a CTL(\*) property that said the lights must come on in a given \textbf{order} then my quotient model is \textbf{invalid}!!
However, something is not quite right...

If I had a CTL(*) property that said the lights must come on in a given order then my quotient model is invalid!!

We need some notion of properties that are invariant under these permutations!
An automorphism group $G$ is an **invariance group** for proposition $p$, or $p$ is **invariant under** $G$ if

$$(\forall \pi \in G)(\forall s \in S)(p \in L(s) \iff p \in L(rep(O(s))))$$

That is, an automorphism group if a proposition holds in a state if and only if it holds it holds in that state in all states in the orbit.
Lemma

Given a

- Kripke structure \((S, R, L)\),
- a set of atomic propositions \(AP\), and
- an invariance (automorphism) group \(G\) on \((S, R, L)\) w.r.t \(AP\),

then the map \(B \subseteq S \times S_G\) where we have \(B(s, O_G(S))\) is a bisimulation between \((S, R, L)\) and \((S_G, R_G, L_G)\).

Corollary

We can model check \((S_G, R_G, L_G)\) instead of \((S, R, L)\)!
Motivation
What is symmetry?
Applications

Model checking with quotient models

To model check these quotient models, we must construct our representation functions and then when searching the state space, we only transition to the representatives and forget the rest of the orbit.
To model check these quotient models, we must construct our representation functions and then when searching the state space, we only \textit{transition} to the \textbf{representatives} and forget the rest of the \textbf{orbit}.

To apply symbolic model checking is more complicated but similar. One constructs a BDD for the representation map $\text{rep} : S \rightarrow S$ and constructs the BDD modelling

- $S_G(x) : = (x = \text{rep}(O(x)))$
- $R_G(x, y) : = \exists x_1 \exists y_1 (R(x_1, y_1) \land \text{rep}(O(x_1)) = x \land \text{rep}(O(y_1)) = y)$
Applications
1 Motivation

2 What is symmetry?

3 Applications
Although we may model check systems using quotient models to reduce the size based on symmetries, there are some problems

- The orbit relation is hard to calculate
Although we may model check systems using quotient models to reduce the size based on symmetries, there are some problems:

- The orbit relation is hard to calculate (graph isomorphism)
Although we may model check systems using quotient models to reduce the size based on symmetries, there are some problems:

- The orbit relation is hard to calculate (graph isomorphism).
- Non-trivial symmetries are hard to detect in the first place!
Although we may model check systems using quotient models to reduce the size based on symmetries, there are some problems:

- The orbit relation is hard to calculate (graph isomorphism)
- Non-trivial symmetries are hard to detect in the first place!
- How do we bring this into the real world? How do we allow non-experts to use this?
Solutions

- Do it all by hand?
Solutions

- Do it all by hand?
- Use *domain specific languages* which know about symmetry
Solutions

- Do it all by hand?
- Use *domain specific languages* which know about symmetry
- Only partially reduce or remove symmetries
Example

Motivation
What is symmetry?
Applications

Lightbulbs!
Example

Lightbulbs!
What is symmetry?

Example

Lightbulbs!
Example

Lightbulbs!
Motivation
What is symmetry?
Applications

Example

Lightbulbs!
Example: DSL with Lightbulbs

You might have:

\[ L : \text{Nat}[3] \times \text{Nat} \]
\[ \text{Ln} : \text{Nat}[3] \times \text{Nat} \]

// One one light bulb at a time
\[ (-L[1] \land -L[2] \land -L[3] \land \text{Ln}[1] \land -\text{Ln}[2] \land L[3]) \lor \]
\[ (-L[1] \land -L[2] \land -L[3] \land -\text{Ln}[1] \land \text{Ln}[2] \land L[3]) \lor \]
\[ \ldots \]
Example: DSL with Lightbulbs

You might have:

$L : \text{Nat}[3] \times \text{Nat}$

$L_n : \text{Nat}[3] \times \text{Nat}$

// One one light bulb at a time


...

but we could write replace “Nat” here with “EquivObj” which the model checker knows has the symmetric group over it. In this way, the user specifying informs the model checker (in a nice way) that the symmetry exists.
Symmetry can reduce complexity *alot*, but... it’s **hard**!

There are some techniques to help make thing simpler... but it’s ongoing.
Thanks!