# cubesProject Summary

Solving the popular kid's toy that asks you to place coloured cubes in a number of slots so that now side has the same colours.

# Propositions

*List of the propositions used in the model, and their (English) interpretation.*

# Constraints

*List of constraint types used in the model and their (English) interpretation. You only need to provide one example for each constraint type: e.g., if you have constraints saying “cars have one colour assigned” in a car configuration setting, then you only need to show the constraints for a single car. Essentially, we want to see the pattern for all of the types of constraints, and not every constraint enumerated.*

# Model Exploration

## Dice Details

At one point, I wasn’t sure what was happening with the dice placed in a slot forcing the case colours to be set correctly. In order to debug this, I wound up implementing a function that printed all of the information about the propositions of a certain dice that happened to be true. This function looked like this:

Text

Description automatically generated

Essentially, we just want to find all of the parts of the solution that deal with some small part of the overall problem. We were able to use this to figure out what was happening with a particular dice, and cross reference it with the visualization used for the entire case.

# Jape Proof Ideas

We will work on a simplified setting for these proofs. For example, fewer cubes, colours, slots, etc.

## All-but-one colour implies the last

***For a side (e.g., “front”), having all but one colour implies that the last colour will be in the final slot.***

Here, we use P<s><c> to represent that “the front side in slot <s> is colour <c>”. Our premises include the following:

* Table

  Description automatically generatedColour can’t appear in two slots:  
  ¬P1R ∨ ¬P2R,  
  ¬P1G ∨ ¬P2G,  
  …
* Every slot has some colour:  
  P1R ∨ P1G ∨ P1B,

P2R ∨ P2G ∨ P2B,

P3R ∨ P3G ∨ P3B,

* First slot is red and second is green:  
  P1R,

P2G

Finally, we want to deduce the final slot colour, P3B. The complete jape proof is shown on the right.

## Can’t fill 3 slots with 2 colours

If we only have two colours modeled, then there’s no way to satisfy things for 3 slots.

This is a long proof of disjunctive case-based reasoning. We use the same propositions as before, except remove any mention of the colour blue. The sequent to prove is…  
¬P1R ∨ ¬P2R, ¬P1G ∨ ¬P2G, ¬P1R ∨ ¬P3R, ¬P1G ∨ ¬P3G, ¬P2R ∨ ¬P3R, ¬P2G ∨ ¬P3G, P1R ∨ P1G, P2R ∨ P2G, P3R ∨ P3G ⊢ ⊥  
…and the proof is as follows:

Table

Description automatically generated

## All-red dice implies blue beside

***If there are just two slots, and an all-red dice is placed in the first slot, then the second must be blue.***

{{{ proof left as an exercise for the reader 😉. **NOTE:** you are not allowed to do the same for your project – doing so (like I have) would mean losing all marks for the final sequent }}

# First-Order Extension

If we were to extend this to first-order logic, then the quantification would mirror the code quite closely – iterating over the cubes, colours, sides, etc. This naturally extends things to an arbitrary number of colours, cubes, etc. An example of some of the constraints in a first-order extension:

* Coming soon…

# Useful Notation

*Feel free to copy/paste the symbols here and remove this section before submitting.*