# Project Summary

*Short summary of the project setting.*

# 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

*List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.*

## Missing TAs

A screen shot of a computer

Description automatically generated

Here, I noticed there were no TAs. It occurred to me that I need a constraint to say how many Tas we need for a particular course. This led me to the following constraint…

A black background with white text

Description automatically generatedA screen shot of a computer program

Description automatically generated

Unfortunately, while this assigned a TA to every course, it had two issues. (1) some had many TA’s, and (2) some had just one (see right). The fix was to identify the exact two TA’s for a course like this:

A computer screen with text and numbers

Description automatically generated

This led to exactly 2 assigned to every course.

## Better Student Pref Display

I was struggling to get a sense of what the preferences led to with respect to the course selection. To overcome this, I made the grid of students x courses prettier, and made the entries of course selections green. This let me very quickly see that every course has 2 Tas, and what their preference was.

A computer screen shot of code

Description automatically generated

## Forcing a Bad Nash

The Nash Equilibrium constraint – that no two students mutually want to swap – is a fairly complex one:

A screen shot of a computer program

Description automatically generated

In order to test that things are working, I changed it ever so slightly so that we force every pair of students assigned to different courses to be violating the Nash Equilibrium:



This led to the following solution:

A black and white grid with green numbers and white text

Description automatically generated

Indeed, we can confirm that every assignment has a better option on every other course.

## Forced Imperfection

To explore the model further, I decided to force sub-optimal preference assignments. Similar to the Nash Equilibrium, the idea is to force *some* other option for every student that looks better. This runs up again the Nash Equilibrium constraint in an interesting way, and force a more complex solution to be computed. This was the constraint specification:

A screen shot of a computer code

Description automatically generated

Note that the options range over all courses, and not just levels on a single course. This is one solution that was computed:

A black board with white text and green numbers

Description automatically generated

Note that every student has another course of a higher preference, but no Nash Equilibrium constraint is violated.

# Jape Proof Ideas

*List the ideas you have to build sequents & proofs that relate to your project.*

1. “If a student isn’t assigned to 2 of 3 courses, they must be assigned to the third.”
2. “If a student is assigned to a course, then it isn’t one with a preference of one.”
3. “If there are 3 students, and a prof ranks one low, then the other two must be ranked highly”

# First-Order Extension

*Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.* ***There is no need to implement this extension!***

## Domain of Discourse

* Natural numbers (used for levels)
* Objects for profs
* Objects for students (TAs)
* Objects for courses

## Predicates

* Assigned(x,y): Student x is assigned to course y for a TA’ship
* Preference(x,y,z): Student x has a preference level of z (1->5) for course y
* ProfPref(w,x,y,z): Prof w has a preference for TA x at level z (1->5) for course y
* MaxGrad(x,y): The maximum number of graduate students TA’ing course x is y
* Assume that we have equality for objects (e.g., or )
* …

We may want to specify the types of individual objects, so that the quantification is a little more oriented to the objects we use. These would be the types for this particular project:

* Student(x): x is a student
* Prof(x): x is a prof
* Course(x): x is a course
* Num(x): x is a number

## Functions

## Constraints

* All TAs can be assigned to only one course  
     
    
  If we want to make sure that the objects are of the correct type, then the formula would be:

Because it makes the formulae unwieldy to always include the types of the objects we quantify over, we will assume that they are implicitly included whenever we have a quantifier.

* No course gets a TA ranked 2 or lower by the instructor
* Some profs can veto certain TAs  
   Interpreted as: a prof can forgo assigning *any* level to a particular TA (across all courses)  
     
    
   If instead, we interpret it as having a value of 1 for the TA
* Nash equilibrium: no swap of TAs/Courses should lead to a better outcome (i.e., more preferences satisfied)

## Theorems

* If a prof didn’t rank a student 1-4, then they must have ranked them 5  
   …

# Useful Notation

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