# Project Summary

Many people know Pokémon for its messages of friendship and its adorable yellow mascot. However, Pokémon also are monstrous beings who we pit against each other for entertainment.

This project- named “GOTTA CATCH ‘EM ALL” aims to determine an estimate of how likely a team of Pokémon will beat another in battle. This model is based on some key assumptions:

1. That you are battling using the online Pokémon Battle Simulator – Pokémon Showdown
   1. So, you would know the opponent’s team’s typing, hp, and speed.
   2. The opponent is likely using the recommended set for their Pokémon.
   3. Because we do not know opponent’s moves, we assume they have an attack of their typing(s).
   4. Accuracy will not be accounted for; I do not have access to the data for all moves ☹.
   5. Cannot account for switching of Pokémon, see Exploring for why.
2. That most Pokémon fall into 3 broad categories of Wall Breaker, Wall, and Sweeper. Note: there are other categories, but they are not possible to implement due to the wide variety and more advanced mechanics.
3. All Pokémon on both teams are level 100s with max IVs.

# Propositions

We have come up with following propositions:

* ***outspeed****:* Validates as true when your Pokémon out speeds the opponent.
* ***resist****:* Which is a probability from 0.25 – 1.00, based on likelihood of your Pokemon surviving opponent’s attack based off their typing.
* ***rand****:* In Pokémon, every time an attack is made, it has a chance of dealing a high roll of damage or low roll. When true, the attack is a high roll. When false, attack is a low roll.
* ***dmg\_high/dmg\_low****:* Corresponds to your Pokémon’s chance of beating the opponent with their attack. Ranges from 0.25-1.00. Example: So, if your attack, when calculated using the attack formula deals 100/200 damage (depending on *rand)*, and your opponent has 400HP, it means *dmg\_high/dmg\_low* respectively evaluate to a probability of 0.25/0.50.
* ***s\_moves****:* Proposition that determines if your Pokémon has at least 2 status moves.

# Constraints

* If your Wall breaker Pokémon beats opponent:
  + ((rand ∧ dmg\_high) ∨ (~rand ∧ dmg\_low)) ∧ (outspeed ∨ resist)
  + Basically, means if damage is enough to beat Pokémon and you out speed or resist (as you need to either deal damage faster, or be able to take the opponents hits).
* If your Wall Pokémon beats opponent:
  + ~ ((rand ∧ dmg\_high) ∨ (~rand ∧ dmg\_low)) ∧ (s\_moves ∨ resist)
  + Basically, means if damage from opponent is not enough to beat your Pokemon, and you either have status moves (which can heal you, or chip away at opponent’s health) or can resist their attacks.
* If your Sweeper Pokémon beats opponent:
  + ((rand ∧ dmg\_high) ∨ (~rand ∧ dmg\_low)) ∧ resist
  + Basically, means if damage is enough to beat Pokémon and you out speed. Sweeper Pokémon need one turn to set up their boosts, so if you can resist the Pokémon’s attack for one turn, you will probably win. In addition, Sweeper Pokémon are usually fast (or will gain speed after their boosts) so the out-speed proposition is unnecessary.
* For each opponent’s Pokémon, the matchup constraints are added to an array. For an opponent’s Pokémon to be considered beatable you must have at least 2 apt counters. This is defined as a constraint as well, however the formula is too long to show. Please see the python file least\_x\_of\_y to see least\_two\_of\_six ().

# Model Exploration

**Testing constraints/Propositions:**

Pokémon is an infamously random game. Chance plays a role in how much damage you deal, and how much damage your opponent deals. I wanted to model this probability using the 204-logic library, and I decided to do so by generating random variables and putting them in ands/ors. For example, if I had a theory that was P ∧ Q, out of the 4 possible models, only 1 would evaluate true. So, I used this principle to create probabilities for damages, and resistance to attacks (see propositions for greater explanations. I made sure to test this by running theories only of these constraints to make sure they returned the expected probability, and it worked.

Throughout my project, I kept debating what should constitute as a beaten Pokémon. I was debating whether it should be just the Pokémon has 1 good counters, or 2 good counters. I settled on 2, as during a battle a few Pokémon may faint, so having 2 counters would constitute a good constraint.

I tweaked the theory throughout by running the model with a strong team vs. a weak team (and vice versa), until it got to a point where the strong team consistently beat the weak (and the weak consistently lost to the strong).

During an actual Pokémon battle, one can switch out their Pokémon. While trying to implement this, I realized it effectively squared my theory size, and number of solutions. To the Point where it took minutes to run (and that was only with 3 Pokémon). It may have also been something about my code being ineffective, but I decided that I would remove the switching option entirely.

**Example Solution Analysis:**

The solutions of my model (meaning chance to beat opponent’s team) vary from 40~75%. This may be due to the model acting as a real person and not being able to determine the potential moves/abilities for the opponent’s Pokémon. Which in turn causes a loss in accuracy. However, I wanted to add some more functionality to the model, so I decided to also list the opponent’s Pokémon that will give you the most trouble.

**Balancing the Model:**

This model is based off a game, and with that unique challenges arise. I found that using certain battle mechanics/Pokémon such as Legendries, Mega-Evolution, Z-Move, and Dynamax all wildly skew the results. For example, using Alakazam against teams without multiple dark types would guarantee a win. Also, the Gen8 rules of Pokémon were unfamiliar with me so I had to use the Gen7 engine. I ultimately did not implement Z-Moves, but one could technically still use Mega or Legendries (not recommended though, as it does skew results).

**The Struggle of Making Moves:**

The most difficult part of making the model was deciding how to model Pokémon attacks, aka “moves”. For my draft I simply made them consist of a type and a category (Physical Attack, Special Attack, Status …) but I found that to be too simple and resulted in some weak Pokémon winning (as it did not incorporate base damage, stab, or other mechanics). I then decided to try and make a proper attack system in code, and then dynamically set constraints based off results. However, I soon realized that trying to code the entire Pokémon attack system would be too long a task. So instead, I decided to use the same idea, but use an external library instead. Only issue with that, however, was that the go to library for the task was a node package. So, I created a node/express API to handle attack calculation and proceeded with my plan to dynamically create constraints based off attack damage!

# First-Order Extension

Extending our project into a predicate logic setting would simplify it. Here are some following examples of how we would simplify the project.

**Variable guide:**

*r:* random damage modifier

*ld:* low damage

*hd*: high damage

*res*: resist opponents attacks

*o:* out speeds opponents

*s:* you have s status/healing move

*ohd*: opponent does high damage

*old*: opponent does low damage

* If each Pokemon on your team were Wall Breakers, then the following formula would suffice:

∀x.((o(x)∧hd(x)∧res(x))→w(x)), *where w is winning against the opponent team.*

* However, a Pokémon can be either a Wall Breaker, Sweeper, or Wall. So, this formula should be used instead:

∀x.(((r(x)∧hd(x))∨((¬r(x)∧ld(x))∧(o(x)∨res(x)))))∨

((((r(x)∧hd(x))∨(¬r(x)∧ld(x)))∧res(x))))∨

((¬r(x)∧ohd(x))∨(¬r(x)∧old(x)))∧(s(x)∨res(x)))) →w(x))*, where w is winning against opponent team.*

This formula would remove the need for loops and the need for checking in code the type of the Pokémon.

* In addition, the at least 2 of 6 formula could be greatly simplified into the following:

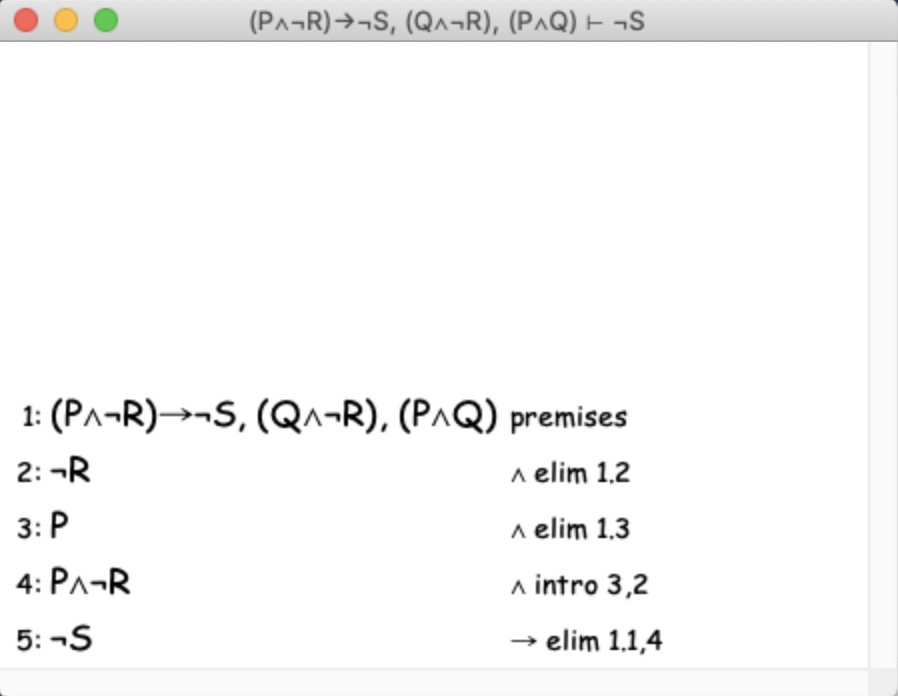
∃x(P(x)∧∃y(P(x)∧y≠x))

We could use the formula, with a domain of {1…6} to make sure that at least 2 of the Pokémon on our team act as a good counter against each of the opponent’s Pokémon.

# Jape Proofs

*The variables used and their meanings are as follows:*

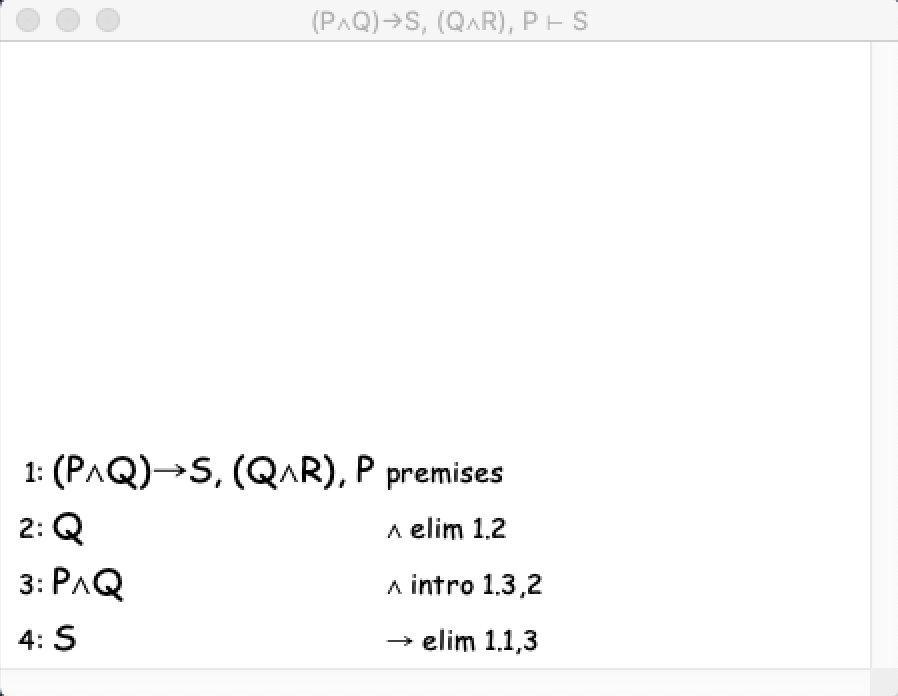
* **P: Your type is effective**
* **Q: Your health is higher**
* **R: Your accuracy is higher**
* **S: You win**
* **¬P: Opponent’s type is effective**
* **¬Q: Opponent’s health is higher**
* **¬R: Opponent’s accuracy is higher**
* **¬S: Opponent’s wins**



*Explanation:*

Hypotheses - Our Pokémon’s type is stronger than the opponent Pokémon’s type and our health is higher than the Pokémon’s health, our health is higher than the opponent Pokémon’s health and the opponent Pokémon’s accuracy is higher than our Pokémon’s accuracy, If our Pokémon’s type is stronger than the opponent Pokémon’s type and the opponent Pokémon’s accuracy is higher than our Pokémon’s accuracy, then the opponent’s Pokémon wins.

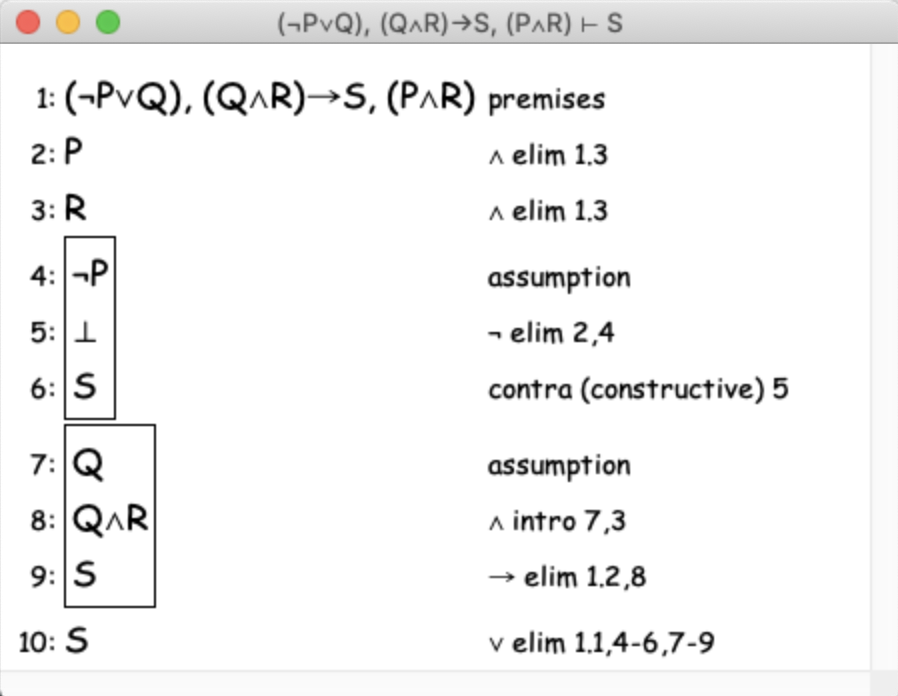
**Conclusion – Opponent’s Pokémon wins.**



*Explanation:*

Hypotheses – Our Pokémon’s type is stronger than the opponent’s type, Our Pokémon’s health is higher than opponent Pokémon’s health and our Pokémon’s accuracy is higher than opponent Pokémon’s accuracy, If our Pokémon’s type is stronger and our Pokémon’s health is higher than opponent Pokémon’s health, then our Pokémon win.

**Conclusion – Our Pokémon wins.**



*Explanation:*

Hypotheses – Our Pokémon’s type is stronger than the opponent’s type and our Pokémon’s accuracy is higher than opponent Pokémon’s accuracy, either our health is higher than opponent Pokémon’s health or the opponent Pokémon’s type is stronger, if our accuracy is higher and our health is higher, our Pokémon win.

**Conclusion – Our Pokémon wins.**