Unplugged Reinforcement Learning Activity

Subject: Computer Science Foundations
Grade Level: 6-12
Date: 10/29/24
Duration: 30-60 mins
Teacher: any

Lesson Title/Topic: Teaching a computer to play checkers: with candy!

Learning Objectives:

- By the end of this lesson, students will be able to:
    - Define the concept and role of reinforcement learning
    - Explain how the “computer” from the activity “learns”

Materials Needed:

- Printouts - game states, checkers board, checkers pieces
- Cups
- Skittles (can be substituted with marbles, beads, etc)

Introduction (5-10 minutes):

- **Hook:** Group discussion - How do humans learn? How does a baby learn? How does a dog learn to sit? Getting a treat is reinforcement. But how do we motivate a computer with no stomach?

Instruction (5-10 minutes):

1. **Explanation of activity:** Rules and materials

2. **Example:** Run through a few rounds of the activity all together.

Activity (10-20 minutes):

- **Activity:** Each group will repeatedly “play” against the “computer” and update the game state (adding or removing skittles) as they go

Conclusion of activity (5-10 minutes):

- **Presentation:** Have groups share what changes they have seen
- **Reflection:** Explain how this type of “learning” is used in computer systems people use every day.

Reflection (Post-Lesson):

- What went well?
- What could be improved?
- Notes for next time:

Example Script for lecture:
Today, we're going to peek behind the scenes at one type of AI. We're going to talk about reinforcement learning. You might see headlines that say something along the lines of "Scientists taught an AI to detect the flu" or "An AI learned to predict football games." But what does that actually mean, for a computer to learn?

Well, that's a hard question. Let's move it a little closer to home. How does a human learn? Well, that's also a tricky one. Let's go a bit simpler: a dog. How do you teach a dog to do a trick? *discuss*
You give it a treat when it does the right thing, like sitting. So, we can use treats to help a dog learn something. But computers don't have stomachs so we'll have to think of something else there. We can't tell a computer to like something, but we can alter it to make decisions based on our feedback. A dog might do something because it likes the reward, and a computer might do something because we've told it that's the best choice of action.

So, we're about to develop our own computer and have it learn how to play tiny checkers. First, I'll give you a refresher on checkers. Each side has two pieces. They start on the light squares and can only move diagonally. There are two ways to win. One: by jumping over your opponent. Two: by reaching the other side. The blue player, our human player, always goes first. Very simple. And we'll talk later about why this example has to be so simple.

Alright, how does this computer work and why in the world do I have so many Skittles in cups? Each of these cups represents a specific state, which is a snapshot of the game in that moment in time. In each state, the computer has a decision to make: what checker do I move and where do I move it? Here's where the skittles come in.
Grabs cup from turn 1
If you see here, each possible move the computer can make matches one of these arrows, and these arrows match the color of the Skittles in the cup. When we get to this game state and need to make our decision, we will close our eyes and randomly pick out a skittle. Now we make the move that matches that color. This skittle is *color* so we make this move here *make the move*

Now, that's how we make choices. But how do we learn? First things first, we need to set this Skittle out to remember what move we made. We are going to learn from our past mistakes or successes. So, let's finish this game real quick.
Have the human move then go through another choice

talk about win first if win, loss first if loss

Allright, we’ve won. Yes! Now we have to tell our system that it did a good job. It should make those moves more often. How could we take this move here and say good job, next time we come to this game state, you should favor that specific move?
discuss
What we can do is put this Skittle back and then add one more of this same color. Now, instead of having an even chance of making each move, the computer will have a bigger chance of making this winning move. So what can we do if the computer is lost and we need to say that was a bad move?
discuss
We can remove that Skittle from the cup. Just don’t put it back. Now, the computer has a smaller chance of making that losing move. Alright, we’ve done it. Our computer has learned! But just a tiny bit. Now we’re going to break up into our groups and train our computer for about twenty minutes. Then we’ll meet back up and talk about it.

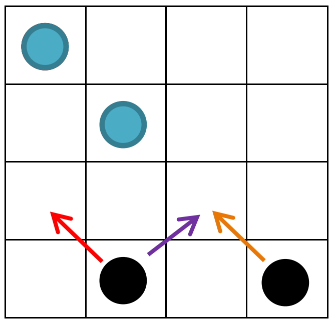
****
group time
****
Welcome back, everyone. I see some very colorful computers. There’s a lot of difference from when we started. Who has a cup that is only one color?
student answer
Nice, now what does that tell us about that color move?
student answer
Yeah, our computer has decided that’s the best move. Whose computer was kinda bad at checkers at the start? I should see all hands. *moment for hands* But that’s ok. Whose computer is kinda good now? *moment for hands* Alright, everyone give yourself a pat on the back for being excellent teachers.

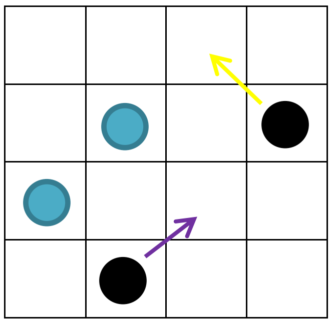
Let’s talk about why we choose checkers. In our game, there are only 3 states at turn one. Then we get way more at turn 2. Imagine how many cups we would have for full sized checkers. Instead of two checkers you would have 8. That’s thousands of cups! Let’s not even mention chess. It’s so many. Even though our game may seem a little silly today, board games have been a big part of computer science for a long time.

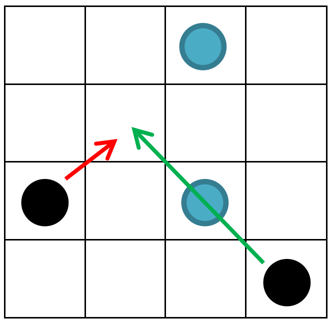
Let’s talk about another game: backgammon. Backgammon is another game with just two players who move pieces on a board. It’s in the same family as checkers and chess. Back in 1992 a man named Gerald Tesauro made his own computer to play and he named it TD-Gammon. He spends years working on this thing, and finally in 1998 he takes TD-Gammon to a tournament. The machine plays against experts and it holds its own. However, all the experts notice - hey, there's one move TD-Gammon is making that's kind of a bad move. No experts would ever make that move. But TD-Gammon was winning lots of games using this move. The computer faces off with the world champion. TD-Gammon just barely loses by 8 points across 100 games. So after the debut of this machine, all the backgammon experts get together and they look at this new move. They try it out, run some calculations, talk about it. And they realize, hey, this is actually a good strategy. Humans have been playing backgammon for over 300 years - and this machine figured out a new strategy! I think that's pretty awesome. At no point did a human tell that machine a strategy. At no point did any of us humans tell our Skittles computer a strategy. But it learned.

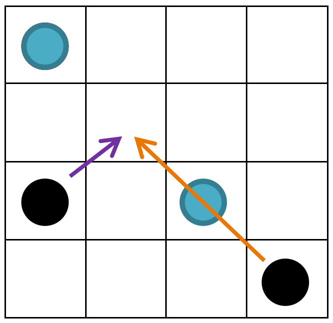
That's pretty cool, but also, what does it have to do with me? Well, this approach, reinforcement learning, is used to train lots of the big AI tools we use today. Reinforcement learning is one method used to train ChatGPT. When you ask ChatGPT a question, imagine how many billions and trillions of Skittles in cups it has! Ok, it's all numbers, but the idea is the same.

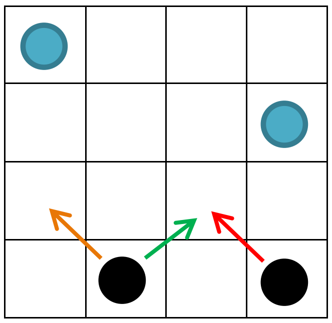
Let's do a little recap of what we've talked about today. Reinforcement Learning is when a computer does an activity over and over and over, and adjusts its decisions based on the feedback of how it did. We did a version of this where our activity was playing checkers, our decisions were making moves, and the computer learns by adding or removing skittles. So, next time you read about or use an AI, think about what we did today. And if you were super duper interested, this is one type of thing that computer scientists do. And you can do it too!

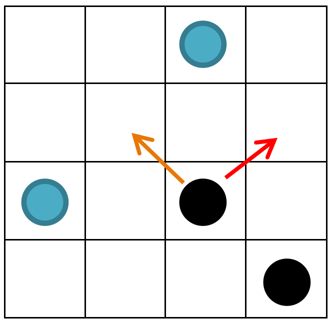

Black's First Turn

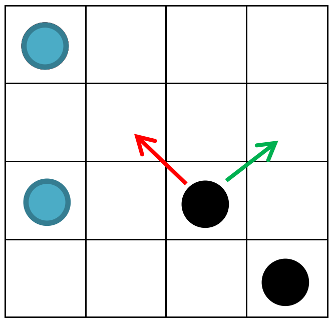

Black's Third Turn

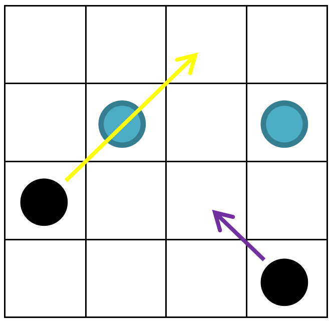

Black's Second Turn

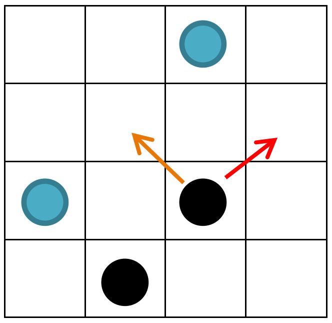

Black's Second Turn

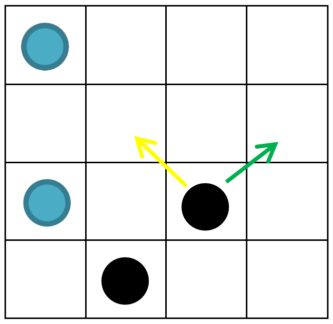

Black's First Turn

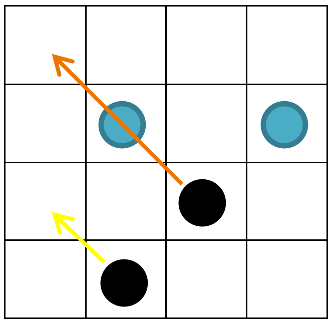

Black's Second Turn

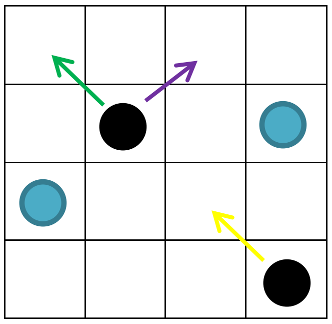

Black's Second Turn

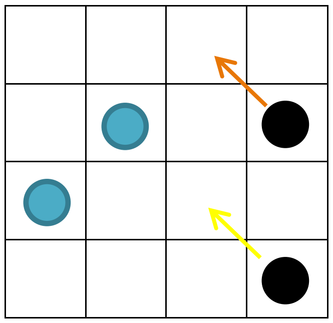

Black's Second Turn

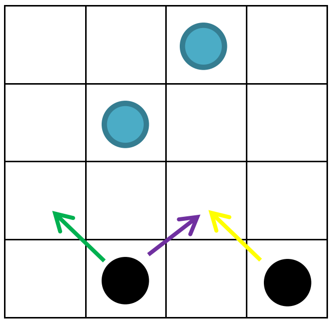

Black's Second Turn

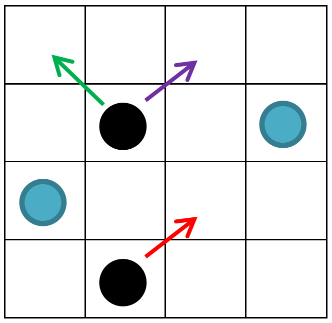

Black's Second Turn

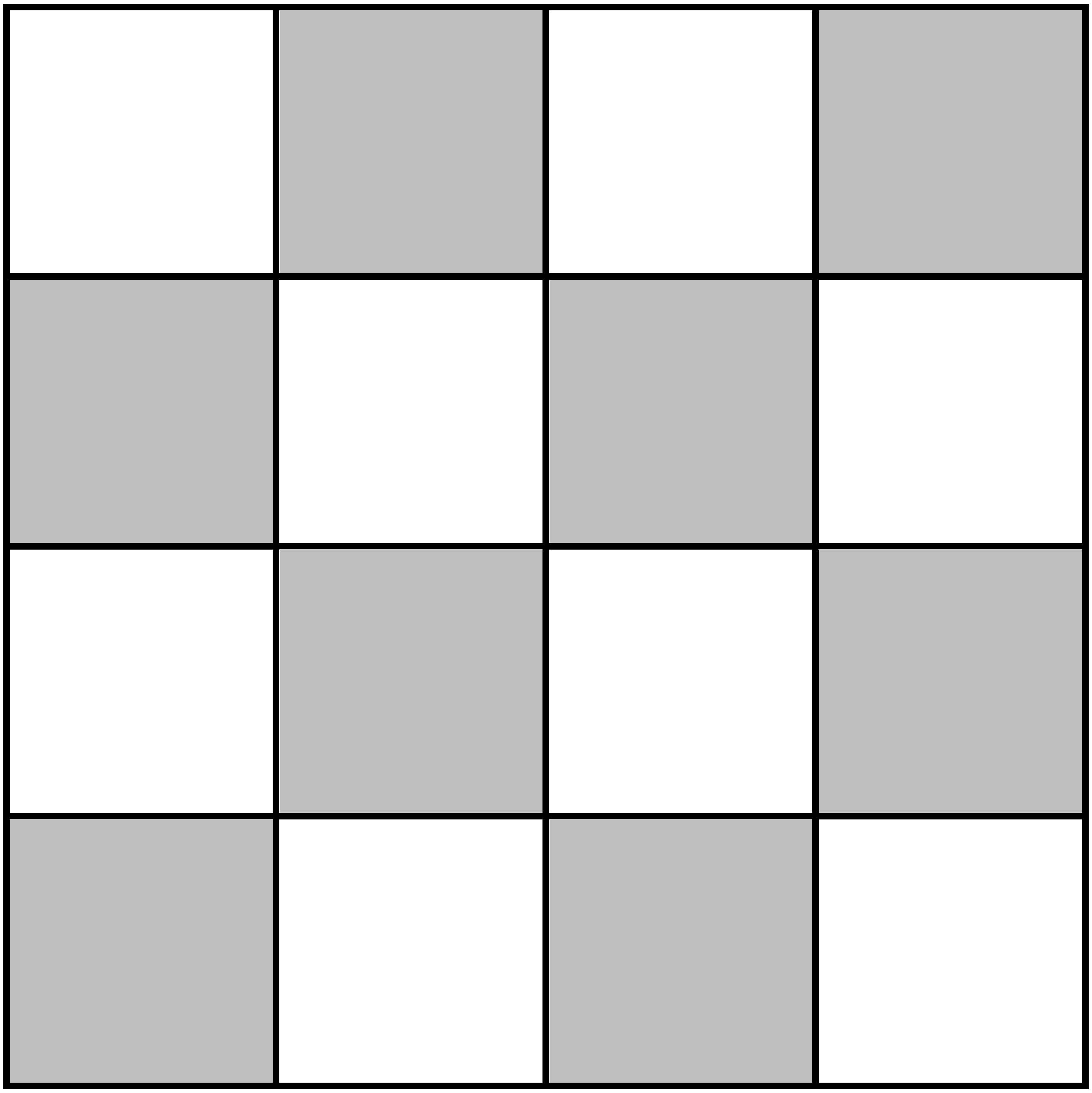

Black's Second Turn


Black's Third Turn

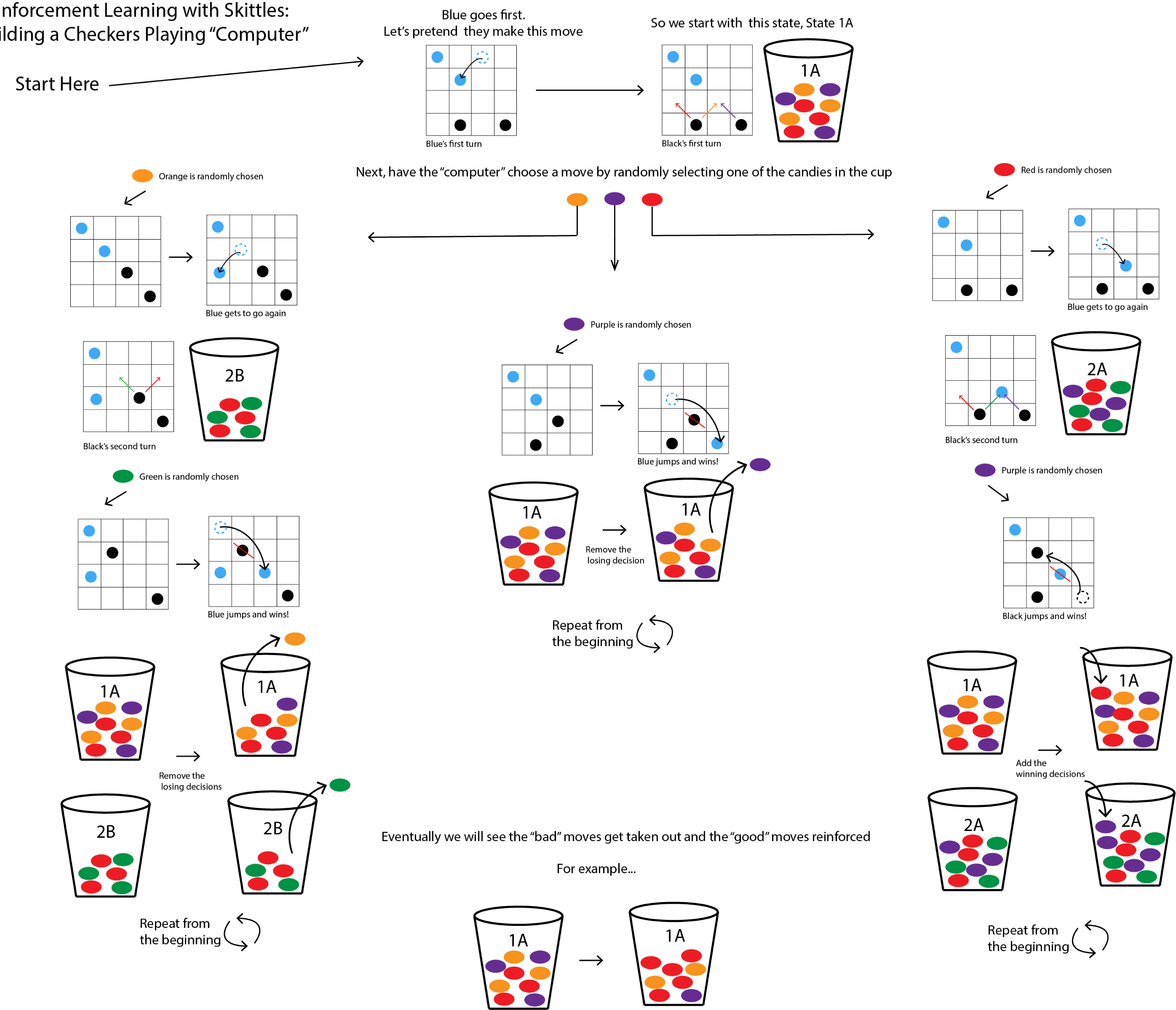

Black's Third Turn


Black's First Turn


Black's Third Turn



Reinforcement Learning with Skittles: Building a Checkers Playing "Computer"


Start Here
Blue goes first.
Let's pretend they make this move
Blue's first turn
So we start with this state, State 1A
Black's first turn
1A
Next, have the "computer" choose a move by randomly selecting one of the candies in the cup
Orange is randomly chosen
Blue gets to go again
Black's second turn
2B
Green is randomly chosen
Blue jumps and wins!
1A
1A
Remove the losing decisions
2B
2B
Repeat from the beginning
Purple is randomly chosen
Blue jumps and wins!
1A
Remove the losing decision
1A
Repeat from the beginning
Red is randomly chosen
Blue gets to go again
Black's second turn
2A
Purple is randomly chosen
Black jumps and wins!
1A
1A
Add the winning decisions
2A
2A
Repeat from the beginning
Eventually we will see the "bad" moves get taken out and the "good" moves reinforced
For example...
1A
1A