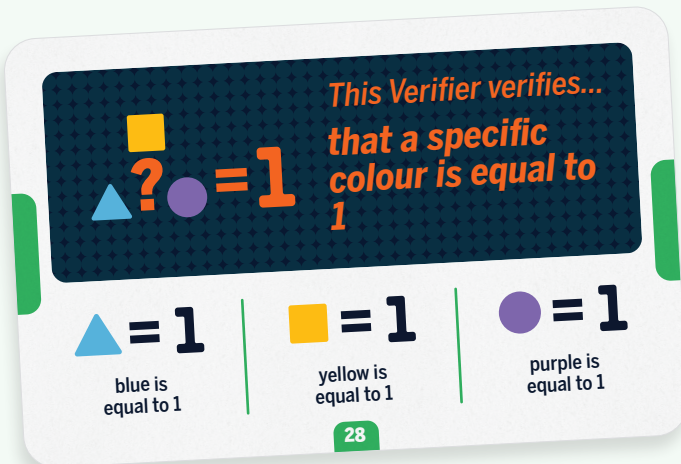# TURING MACHINE

THE 'X' PARADOX

Dear Seasoned Turing Machine Players: If you consider yourself a deduction ninja, you have probably thought to yourself, as you tackle the game's most complicated problems, "hold on a second, the machine is broken!" Computer geniuses, however, have a saying: The Machine is **ALWAYS** Right.

> You have probably come across what we call **'The 'X' Paradox'.**

The 'X' Paradox is the *false interpretation of an* ❌ *answer*. This happens when numbers or criteria that should not have been eliminated were, in fact, eliminated during deduction.

Let us explain with a first example:

This Verifier verifies... that a specific colour is equal to 1

△ = 1 — blue is equal to 1 | 🟨 = 1 — yellow is equal to 1 | 🟣 = 1 — purple is equal to 1

28

Let's imagine that:

- The criterion that will pass this Verifier's test is '**yellow** is equal to 1'
- The final target code is 111.
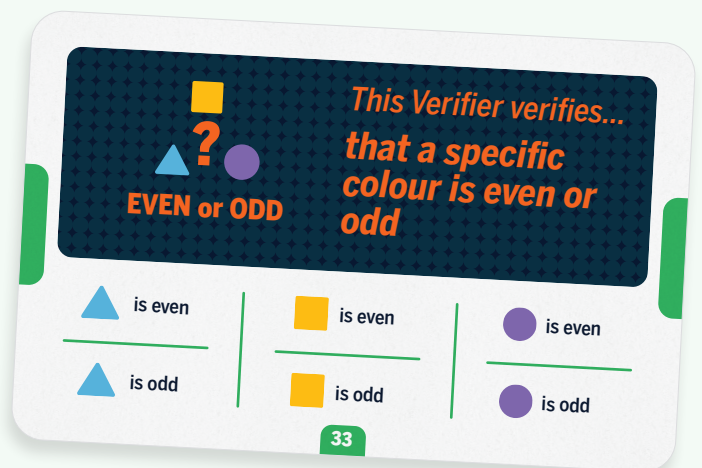
🟨 = 1 — yellow is equal to 1

If I use 1️⃣4️⃣1️⃣ as a proposal, I will get an ❌ because **yellow is not equal to 1 in my proposal.** I am not respecting the criterion required to pass this Verifier's test.

It is NOT telling me that blue or purple are not equal to 1. I got an ❌ **answer even though my blue and purple numbers are in the final code!** This happens because that is not what this Verifier is looking at. In fact, it doesn't even see the purple or blue numbers!

Getting an ❌ answer allows me to eliminate criteria from the list of **THIS CARD ONLY.**

> When a Verifier gives me an ❌ answer, it is not saying "Your numbers are not in the final code", it is saying "You haven't done what I want." **The Verifiers know their criterion, but they don't know the final code.**

Let's apply this same rule to a more advanced card:

This Verifier verifies... that a specific colour is even or odd

EVEN or ODD

△ is even / △ is odd | 🟨 is even / 🟨 is odd | 🟣 is even / 🟣 is odd

33

If I submit 3️⃣3️⃣3️⃣ as my proposal, I am testing the 3 following criteria: Blue is odd, yellow is odd, and purple is odd. I get an ❌.

All I have learned is that to satisfy **THIS** Verifier, I can eliminate these 3 criteria (blue is odd, yellow is odd, and purple is odd). **But in the final code, blue can still be odd!**

△ is even / ~~is odd~~ | 🟨 is even / ~~is odd~~ | 🟣 is even / ~~is odd~~

With this information, **nothing is preventing the final code from being 323.** I didn't learn that blue is not odd in the final code.

If we imagine that the criterion that will pass this Verifier's test is '**yellow** is even', I would get a ✅ if I proposed 3️⃣2️⃣3️⃣. We can see even more clearly in this example that the Verifier doesn't even see the blue and purple numbers in the proposal!