```
+--------------------------+
| CS 4110 - 12/05/2025     |
| Lab 37 - ProgramSynthesis |
+--------------------------+
```

This lecture is based on the FMCAD '16 paper "On ∃ ∀ ∃! Solving: A Case Study on Automated Synthesis of Magic Card Tricks" by Susmit Jha, Vasu Raman (Cornell PhD '13), and Sanjit Seisha: https://people.eecs.berkeley.edu/~sseshia/pubdir/magic-fmcad16.pdf.

We started the lab by doing a "Baby Hummer" card trick. See Chapter 1 of "Magical Mathematics" (Princeton University Press) for details: https://assets.press.princeton.edu/chapters/s9510.pdf.

Why does the trick work?

Let's give names to the various actions we performed on the deck:
* F1: Take the top card and flip it over.
* F2: Take the top two cards and flip them over (i.e., as a bundle).
* C4: Cut the top 4 cards off the deck and put them on the bottom.
* C3: Cut the top 3 cards off the deck and put them on the bottom.
* C2: Cut the top 2 cards off the deck and put them on the bottom.
* C1: Cut the top 1 card off the deck and put it on the bottom.
* Noop: Do nothing.

Observe that C4 = Noop.

Let's write D if a card is facing down and U if it is facing up.

Suppose we start with a 4-card deck in a random order, but with the following orientations: D-U-D-U. That is, the top card, is down, the second card is up, the third card is down, and the last card is up.

Observe what various actions do when applied to the deck:

  F2: D-U-D-U -> D-U-D-U

  C2: D-U-D-U -> D-U-D-U
  C2; F2: D-U-D-U -> D-U-D-U

  C4: D-U-D-U -> D-U-D-U
  C4; F2: D-U-D-U -> D-U-D-U

  C1: D-U-D-U -> U-D-U-D
  C1; F2: D-U-D-U -> U-D-U-D

  C1: D-U-D-U -> U-D-U-D
  C3; F2: D-U-D-U -> U-D-U-D

So in general, for any N in { 1 .. 4 } we have

```
  CN; F2: D-U-D-U -> D-U-D-U
  or
  CN; F2: D-U-D-U -> U-D-U-D
  or
  CN; F2: U-D-U-D -> U-D-U-D
  or
  CN; F2: U-D-U-D -> D-U-D-U
```

That is, no matter how we cut the deck, we always preserve the property of alternating orientations!

Now consider the baby Hummer trick. Let B be the bottom card. The opening does the following:

```
  C1; F1: D-D-D-B -> D-D-B-D -> U-D-B-D
```

Now we do CN;F2 an arbitrary number of times, that is (CN;F2)*. This leaves us in a configuration

```
- B-D-U-D
- U-D-B-D
- D-B-D-U
- D-U-D-B
```

Then we do the closing sequence: F1; C1; C1; F1. Write B' for the original-bottom card flipped to up.

```
- B-D-U-D -> D-D-B'-D
- U-D-B-D -> B'-D-D-D
- D-B-D-U -> D-U-U-B'
- D-U-D-B -> U-B-U-U
```

In all cases, the single card whose orientation is different is the original bottom one.

The rest of the lab developed a naive (i.e., based on enumeration) program synthesizer to generate new card tricks. The main idea was to develop a DSL for actions, and then search for "programs" consisting of sequences of actions that satisfy a given specification -- e.g., for Baby Hummer, that the original bottom card is the only one with a different orientation in the final state. See the accompanying OCaml code...