Exercise 0: (setting up)

1. Make sure you have run

   ```
   python unit_tests.py
   ```

   successfully.

2. Make a new folder called exercises in the main folder of Diddy, and put the file exercise1.diddy we provide in this folder.

3. You can use this file as the basis for the first exercise.

   ```
   python diddy.py exercises/exercise1.diddy
   ```

   should run successfully, but not do anything, as everything in the file is initially commented out:

   ```
   -- (anything after two hyphens is ignored)
   ```

Exercise 1: (defining an SFT) The golden mean shift $X_{\texttt{gm}}$ is the set of all maps $x\colon \mathbb{Z}^2 \to \{0,1\}$ with the property that no pair of symbols 1 occur adjacent to each other. Namely

$$X_{\texttt{gm}} = \{x \in \{0,1\}^{\mathbb{Z}^2} : \text{ for every } u \in \mathbb{Z}^2, \text{ if } x(u) = 1 \text{ then } x(u+(0,1)) = 0 \text{ and } x(u+(1,0)) = 0\}.$$

1. Define the golden mean shift on Diddy. In order to do this, you can use the shorthand

   ```
   %SFT goldenmean Ao <logic expression>
   ```

   where "logic expression" should be replaced by a logical constraint that encodes the condition that defines the golden mean shift. The logical relations in Diddy are written as:

   - implication `->`
   - and `&`
   - or `|`
   - not `!`
   - equivalence `<->`

   Recall that o.rt, o.lt, o.up, o.dn can be used to move around in the grid.

2. Determine if the golden mean shift is empty, for that use the command

   ```
   %empty goldenmean
   ```

3. Play with the tiler to produce windows in the golden mean shift using the command:

```
%tiler goldenmean
```

You can click on nodes to change the symbols. Write 0s by pressing "1", 1s by pressing "2", and "unknown" by pressing u. To erase cells completely, press backspace. Press space to tile.

4. Define the SFT onlyzero which consists of a single configuration with 0s and the SFT onlyone having a single configuration with 1s.

```
%SFT onlyzero Ao ...
%SFT onlyone Ao ...
```

Now try the following commands and interpret the results.

```
%equal goldenmean goldenmean
%contains goldenmean onlyzero
%equal goldenmean onlyzero
%contains goldenmean onlyone
```

5. Define the alternative golden mean shift as the one where the condition is instead given by

$$\text{For every } u \in \mathbb{Z}^d \text{ and } k \in \{(0,1),(0,-1),(-1,0),(1,0)\},\ \text{if } x(u) = 1 \text{ then } x(u+k) = 0.$$

Define it in Diddy and call it "altgoldenmean".

6. What's the difference between the golden mean shift and its alternative version? let's ask Diddy, try the command

```
%equals goldenmean altgoldenmean
```

7. Define the inverse golden mean shift as the one where the roles of 0 and 1 are inverted, that is,

$$X_{\texttt{igm}} = \{x \in \{0,1\}^{\mathbb{Z}^2} :\ \text{for every } u \in \mathbb{Z}^2,\ \text{if } x(u) = 0 \text{ then } x(u+(0,1)) = 1 \text{ and } x(u+(1,0)) = 1\}.$$

Define the inverse golden mean shift in Diddy using the command

```
%SFT invgoldenmean Ao <logic expression>
```

8. Try to figure out what is the intersection of the golden mean shift and the inverse golden mean shift. In Diddy, you can compute the intersection of two SFTs using the command

```
%intersection <name> <SFT1> <SFT2>
```

Compute the intersection of both shifts and use the tiler to verify your intuition.

9. Define the checkerboard shift which consists of the two configurations x and y in $\{0,1\}^{\mathbb{Z}^2}$ which satisfy that $x(m,n) = m + n \bmod 1$ and $y(m,n) = m + n + 1 \bmod 1$. Use Diddy to verify that int is precisely the checkerboard shift.

Exercise 2: (defining a cellular automaton) The 3-dot shift (also called Ledrappier shift) is the set of all maps $x\colon \mathbb{Z}^2 \to \mathbb{Z}/2\mathbb{Z}$ such that for every $u \in \mathbb{Z}^2$ we have

$$x(u) + x(u + (0,1)) + x(u + (1,0)) = 0.$$

1. Define the 3-dot shift in Diddy. Call it `led`.

2. Use the tiler to figure out how configurations in the 3-dot shift look like. In particular, play with border conditions where there is a single 1 in the bottom right corner and 0s in all remaining positions in the south and right borders.

3. The 3-dot shift can also be seen as the *spacetime subshift* (the SFT of all spacetime diagrams) of the CA $\phi\colon (\mathbb{Z}/2\mathbb{Z})^{\mathbb{Z}} \to (\mathbb{Z}/2\mathbb{Z})^{\mathbb{Z}}$ given by the rule

   $$\phi(x)(k) = x(k) + x(k+1) \bmod 2.$$

   Define the automata above using the elementary relations (and, not, or, implication, etc). Use the command

   ```
   %dimension 1
   %CA xor
   0 1 Ao o != o.rt
   ```

   Here "0 1 Ao" means "write 1 if the condition on the right holds": 0 is the name of the unique node, 1 is the symbol written under the condition, and Ao names the variable o used to refer to positions, just like with SFTs. Elsewhere the rule will write 0 by default.

4. Try the command

   ```
   %spacetime st xor
   ```

   This defines st as the spacetime subshift of the automaton xor. Now try

   ```
   %equal led st
   ```

   It should say that they are not equal (which is weird).

5. Try now the following commands

   ```
   %contains led st
   %info led
   %info st
   ```

   What is the issue with the previous exercise?

6. On the GitHub page, figure out a suitable flag for the spacetime command to fix this issue. Confirm that then that the Ledrappier subshift and st ARE equal.

Exercise 3: (counting)

1. Use the command "%topology king" to specify that we use the "king grid" with also diagonal edges (but still just one node 0).

2. Define an analog of the golden mean shift for the king grid. Directions are now called E, NE, N, NW, W, SW, S, SE. Test your golden mean shift in the tiler.

3. The counting quantifier # allows you to count the number of neighbors of a positional variable, which satisfy some expression. Specifically, the expression

   `letnum nbrs := #n[o1] <expression using n> in <expression2>`

   binds the variable nbrs to the number of neighbors n of o at distance at most 1, which satisfy <expression using n>, and then evaluates <expression2> with this binding. For example

   `letnum origins := #n[o2] n@o in <expression2>`

   binds the variable origins to the number 1 (because @ checks for equality of positions, so n@o only happens once) and evaluates <expression2> with origins now meaning 1. Use the counting quantifier to give an alternative definition of the golden mean shift, by saying that if the origin contains 1, then the number of 1-neighbors is 0 (note that the variable n will also range over the origin, so be careful). Verify with the equals command that this defines the same golden mean shift you defined above.

4. For a set of numbers N, we can define the two-dimensional binary subshift $X_{k,N}$ where in each k-by-k box the number of 1s is in the set N. If $n \in \mathbb{N}$ write $X_{k,n} = X_{k,\{n\}}$ for short. Verify with Diddy that $X_{3,n}$ is nonempty for $n = 0, ..., 9$ but not for $n = 10$. Can you prove this by hand?

5. Verify with Diddy that $X_{5,10} \cap X_{3,\{3,4\}}$ is nonempty, but $X_{5,10} \cap X_{3,\{4,5\}}$ is empty.

6. an n-dimensional grid with the ℓ_∞ metric is obtained with the command %topology king$n$. Try experiments like the above in higher dimensions.

7. We can also define a variant of the above subshift where we count 1s in ℓ_1-balls of radius k. In Diddy, write simply %dim n to get the n-dimensional (hyper)grid topology. Let the d-dimensional such subshift be called $Y^d_{k,N}$. The strong Golomb-Welch conjecture states that $Y^d_{k,\{1\}}$ is empty for $d \geq 3, k \geq 2$. Explore the cases $d \in \{2, 3\}, k = 2$ in Diddy. If you have a supercomputer, you can try $d = 4, k = 2$. (To our knowledge, the case $d = 6, k = 2$ is open.)

Exercise 4: (more elaborate example) Consider the Cayley graph of $\mathbb{Z}^2$ given by the generators $S = \{(0, \pm 1), (\pm 1, 0)\}$. That is, the graph where the vertices are given by $\mathbb{Z}^2$ and two vertices v, u are connected by an edge if there is $s \in S$ such that $u + s = v$. The even shift $X_{\texttt{even}}$ is the set of all maps $x \colon \mathbb{Z}^2 \to \{0, 1\}$ with the property that every finite connected component of $x^{-1}(1)$ has an even number of vertices.

1. Prove (on paper!) that the even shift is NOT a subshift of finite type.
2. Show that there exists an alphabet A, an SFT $Y \subset A^{\mathbb{Z}^2}$ and a map $t \colon A \to \{0, 1\}$ such that

   $$x \in X_{\texttt{even}} \text{ if and only if there exists } y \in Y \text{ such that } x(u) = t(y(u)) \text{ for every } u \in \mathbb{Z}^2.$$

   Hint: The handshaking lemma in graph theory tells you that the sum of the degrees of all vertices in a finite graph is equal to twice the number of edges. Define Y as the SFT which encodes a partition of $\mathbb{Z}^2$ in subgraphs where every vertex has odd degree or degree zero.
3. Search for the file `even.diddy` in the examples folder. Try to understand its implementation
4. Try a few commands modifying the definition of `even.diddy`, such as `empty` and `tiler`.

Exercise 5: (have fun)

1. Look through the Wiki to see what the current (documented...) features of Diddy are.
2. Try computing forbidden patterns or a minimal density for your example of choice.
3. Come up with a comparison problem for two-dimensional SFTs, which Diddy cannot solve.

Finally, if you have interesting solutions to some of the exercises, the authors are happy to see them :) If you found bugs, have feature requests, or simply think Diddy is not user-friendly enough, we are also more happy to hear your angry complaints! Diddy should be usable without a computer science degree, and if it is not, it is only because the authors do not have computer science degrees. The authors are Ville Salo (vosalo@utu.fi) and Ilkka Törmä (iatorm@utu.fi). Ilkka could not make it to the school, so Sebastián Barbieri (sebastian.barbieri@usach.cl) helped make today's session possible.