SLUGS

SLUGS: SmalL bUt Complete GROne Synthesizer

- Slugs is a stand-alone reactive synthesis tool for generalized reactivity(1) synthesis.
- Free, open source and available at: https://github.com/VerifiableRobotics/slugs

Using SLUGS

Step 1: Model the 2-player game in .structuredslugs format

Step 2: Convert the file to .slugsin using: slugs/tools/Struc../compiler.py
$ python slugs/tools/Struc../compiler.py [filename].structuredslugs

Step 3: Run slugs with .slugsin file as input and the option you need
$ slugs [filename.slugsin] --explicitStrategy --jsonOutput > [output_filename]

The Structured Slugs Language

Variable Definitions:

```
[INPUT]
a
b:0...10

[OUTPUT]
c:2...8
d
```

Safety Formula G (...):

```
[ENV_TRANS]
a -> (a' <-> ! a)
b' = b + 1

[SYS_TRANS]
d -> (c' = 3)
```

Initial conditions:

```
[ENV_INIT]
! a
b = 1

[SYS_INIT]
d
c = 4
```

Conjunction of Liveness Formulas G F (...):

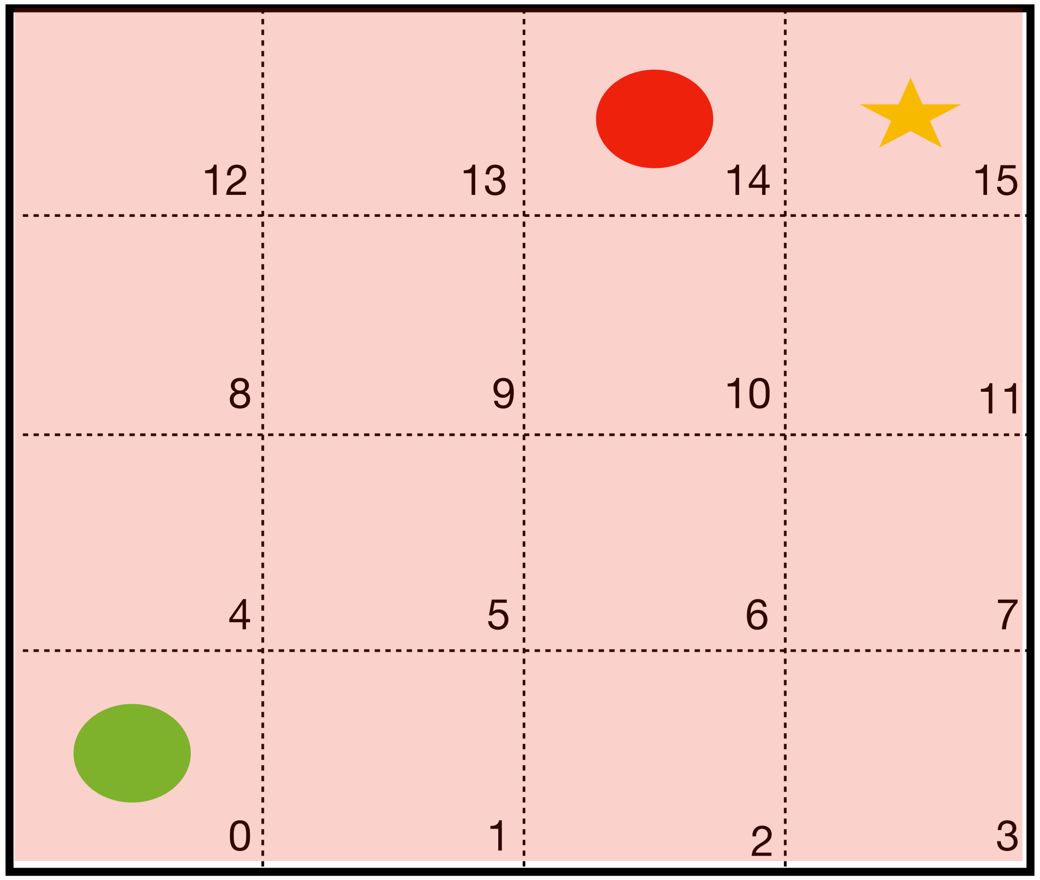
```
[ENV_LIVENESS]
! a | (b = 3)

[SYS_LIVENESS]
d
c = 2
```

SLUGS Example

Model all the allowed transitions of the agent and the obstacle in LTL

```
[ENV_TRANS]
o_state = 0 -> (o_state' = 1) | (o_state' = 4) | (o_state' = 0)
o_state = 4 -> (o_state' = 1) | (o_state' = 8) | (o_state' = 4) | (o_state' = 5)
.
.
a_state = 0 -> (a_state' = 1) | (a_state' = 4) | (a_state' = 0)
a_state = 4 -> (a_state' = 1) | (a_state' = 8) | (a_state' = 4) | (a_state' = 5)
```



The agent must reach the starred location and never crashes into the moving obstacle.

SLUGS Example

Model all the allowed transitions of the agent and the obstacle in LTL

```
[ENV_TRANS]
o_state = 0 -> (o_state' = 1) | (o_state' = 4) | (o_state' = 0)
o_state = 4 -> (o_state' = 1) | (o_state' = 8) | (o_state' = 4) | (o_state' = 5)
.
.
a_state = 0 -> (a_state' = 1) | (a_state' = 4) | (a_state' = 0)
a_state = 4 -> (a_state' = 1) | (a_state' = 8) | (a_state' = 4) | (a_state' = 5)
```

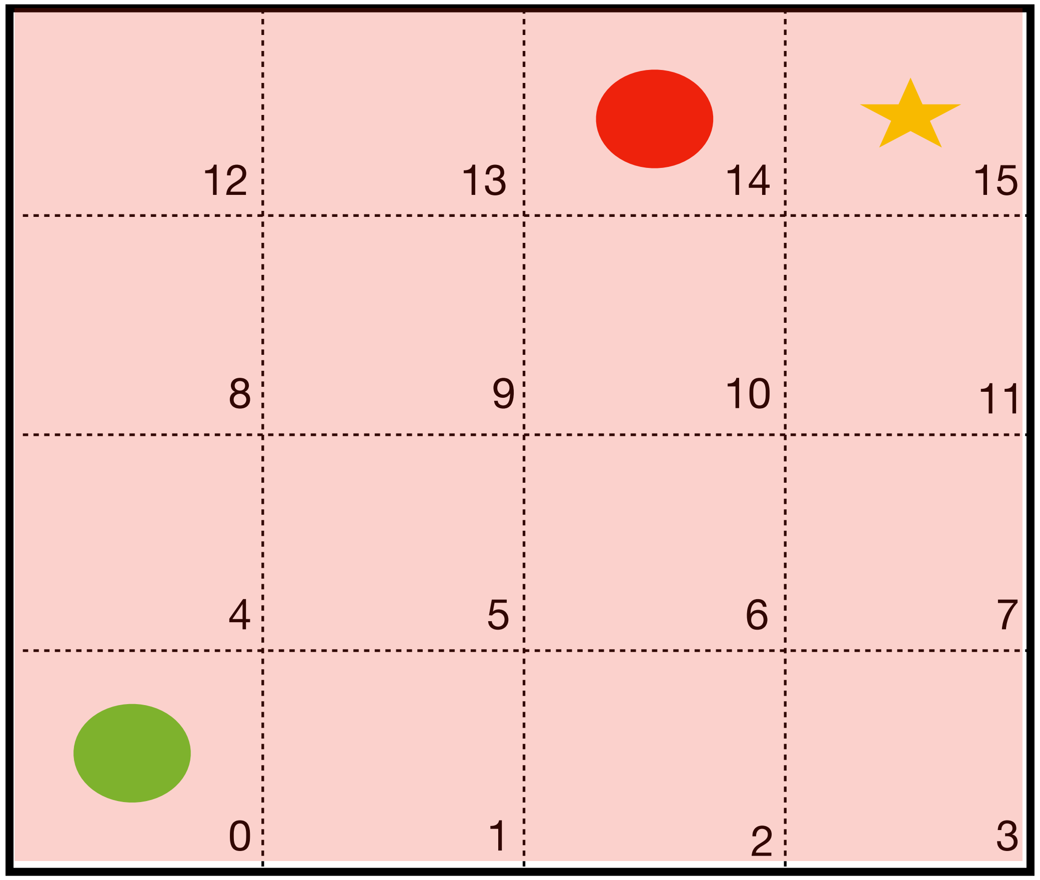
Encode the properties as either safety or liveness LTL specifications

the agent must visit locations 15 and 0 infinitely often

```
[SYS_LIVENESS]
a_state = 15
a_state = 0
```

the agent and obstacle must never crash

```
[SYS_TRANS]
o_state != a_state
o_state' != a_state
o_state != a_state'
```



The agent must reach the starred location and never crashes into the moving obstacle.

SLUGS Example

Model all the allowed transitions of the agent and the obstacle in LTL

```
[ENV_TRANS]
o_state = 0 -> (o_state' = 1) | (o_state' = 4) | (o_state' = 0)
o_state = 4 -> (o_state' = 1) | (o_state' = 8) | (o_state' = 4) | (o_state' = 5)
.
.
a_state = 0 -> (a_state' = 1) | (a_state' = 4) | (a_state' = 0)
a_state = 4 -> (a_state' = 1) | (a_state' = 8) | (a_state' = 4) | (a_state' = 5)
```

Encode the properties as either safety or liveness LTL specifications

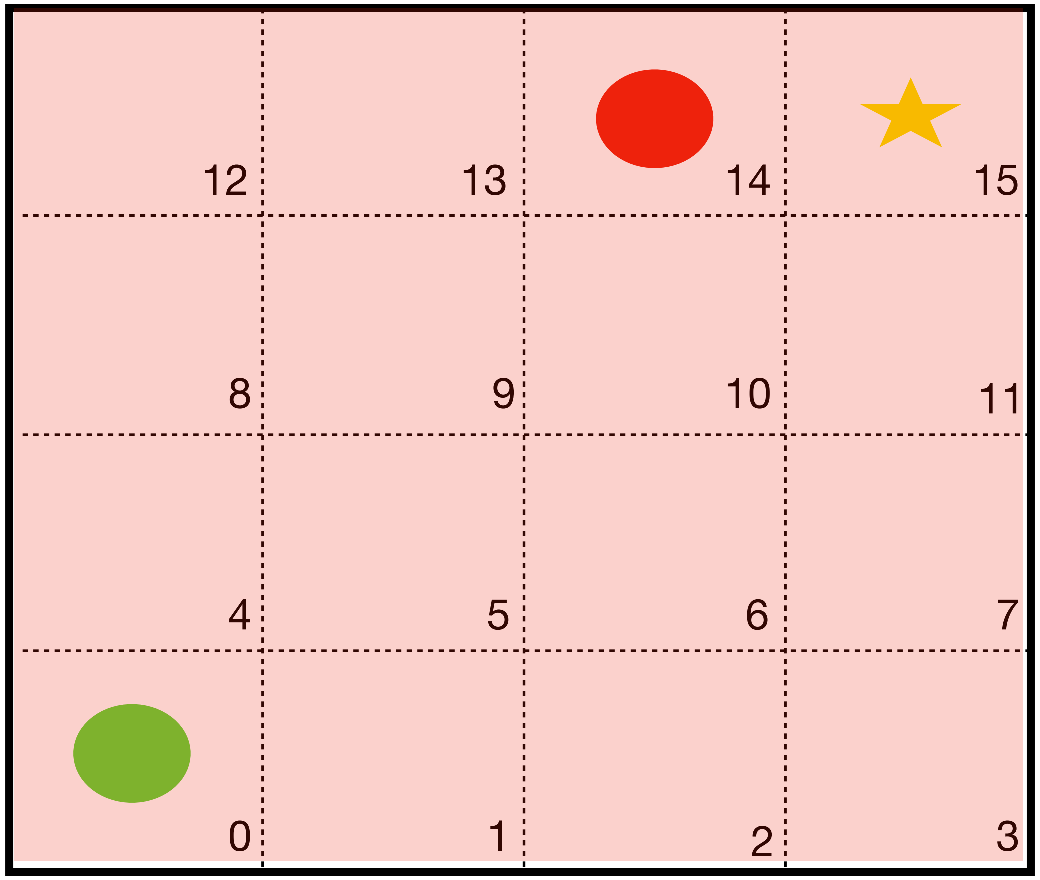
the agent must visit locations 15 and 0 infinitely often

```
[SYS_LIVENESS]
a_state = 15
a_state = 0
```

the agent and obstacle must never crash

```
[SYS_TRANS]
o_state != a_state
o_state' != a_state
o_state != a_state'
```

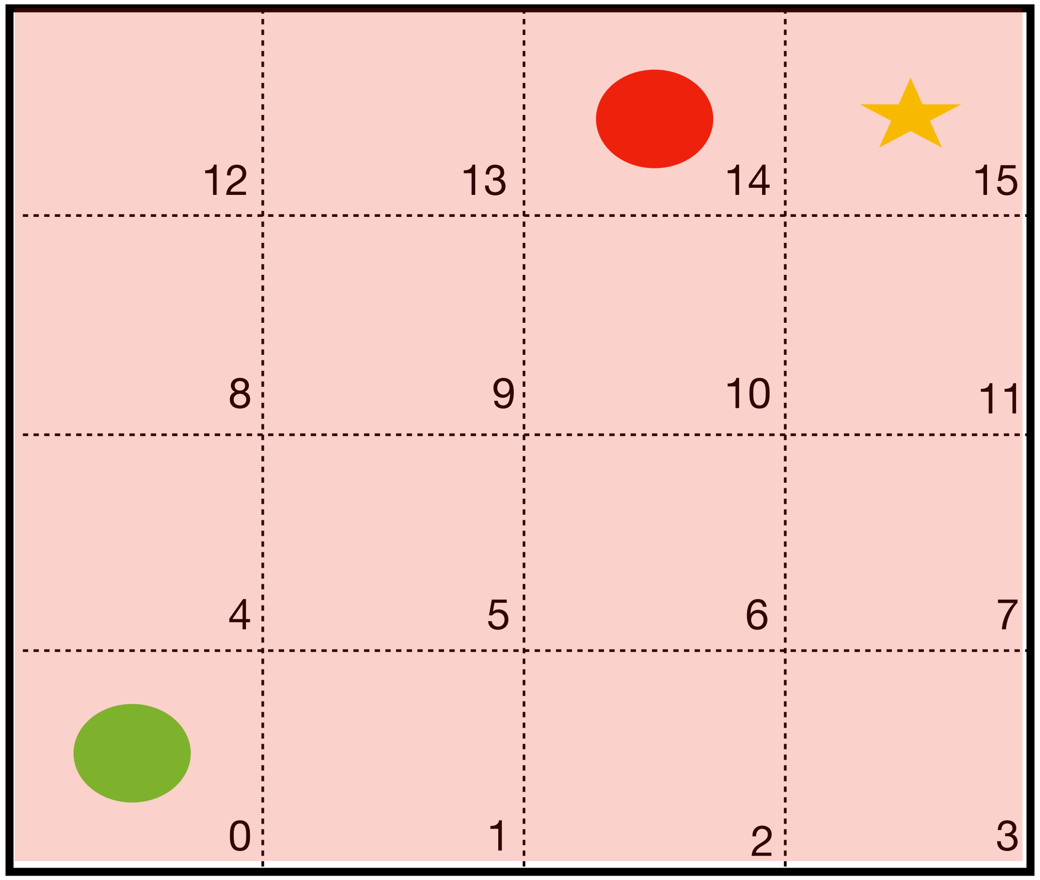
This problem is unrealizable



The agent must reach the starred location and never crashes into the moving obstacle.

SLUGS Example

Incrementally add reasonable assumptions on the environment until it's realizable



The agent must reach the starred location and never crashes into the moving obstacle.

SLUGS Example

Incrementally add reasonable assumptions on the environment until it's realizable

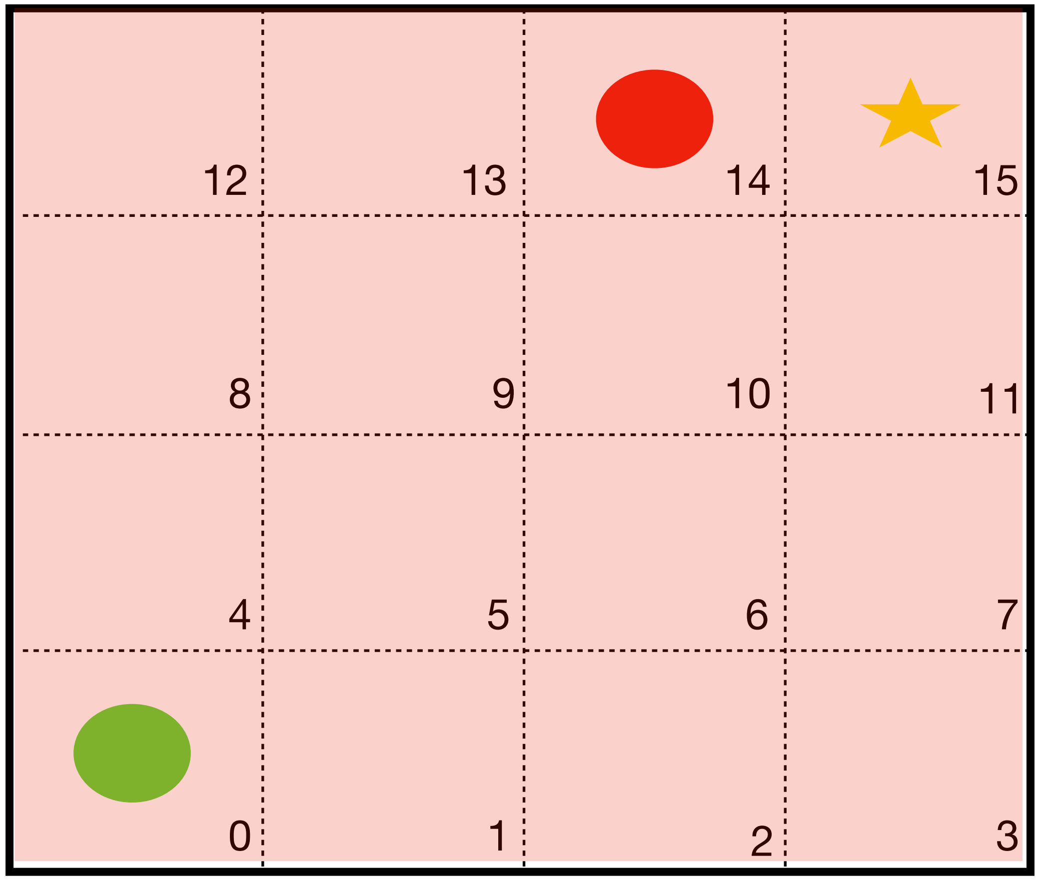
Example:

assume that the moving obstacles visits locations 0 and 15 infinitely often

$G\ (F\ (o\_state = 15) \wedge F\ (o\_state = 0))$

=

[ENV_LIVENESS]
o_state = 15
o_state = 0



Now we can synthesize a reactive controller for this problem

The agent must reach the starred location and never crashes into the moving obstacle.