Logics for Safe AI 2024/2025

Coursework 1

Aran Montero (9540318)
Pablo Pardos (843586)
Jesse Hoiting (4443306)

2.1 Define the system formally

To formally define the system we need to first define three components:

- A set St representing the possible states.
- The transition function $\rightarrow$ with its respective possible actions.
- The valuation function, which labels states with true propositions.

2.1.1 The set of states (St)

There are 4 states:

$$St = \{s_1, s_2, s_3, s_4\}$$

Each state is described as:

$$\begin{aligned}
s_1 &: (\text{surface: True, open: False, sunk: False}) \\
s_2 &: (\text{surface: True, open: True, sunk: False}) \\
s_3 &: (\text{surface: False, open: False, sunk: False}) \\
s_4 &: (\text{surface: False, open: True, sunk: True})
\end{aligned}$$

2.1.2 Transition function

The possible actions and transitions between states are:

$$
\begin{aligned}
&s_1 \xrightarrow{\text{open}} s_2, \quad && s_1 \xrightarrow{\text{close}} s_1, \\
&s_1 \xrightarrow{\text{up}} s_1, \quad && s_1 \xrightarrow{\text{down}} s_3, \\
&s_2 \xrightarrow{\text{close}} s_1, \quad && s_2 \xrightarrow{\text{up}} s_2, \\
&s_2 \xrightarrow{\text{down}} s_4, \quad && s_2 \xrightarrow{\text{open}} s_2, \\
&s_3 \xrightarrow{\text{up}} s_1, \quad && s_3 \xrightarrow{\text{down}} s_3, \\
&s_3 \xrightarrow{\text{open}} s_4, \quad && s_4 \xrightarrow{\text{close}} s_3, \\
&s_4 \xrightarrow{\text{down}} s_4.
\end{aligned}
$$

2.1.3 Valuation function

The valuation for each state is:

$$
\begin{aligned}
&V(s_1) \to \{\text{surface}\}, \\
&V(s_2) \to \{\text{surface, open}\}, \\
&V(s_3) \to \{\}, \\
&V(s_4) \to \{\text{open, sunk}\}.
\end{aligned}
$$

2.1.4 Visual representation

As a way to visualize it we decided to create a graphical representation of this model:

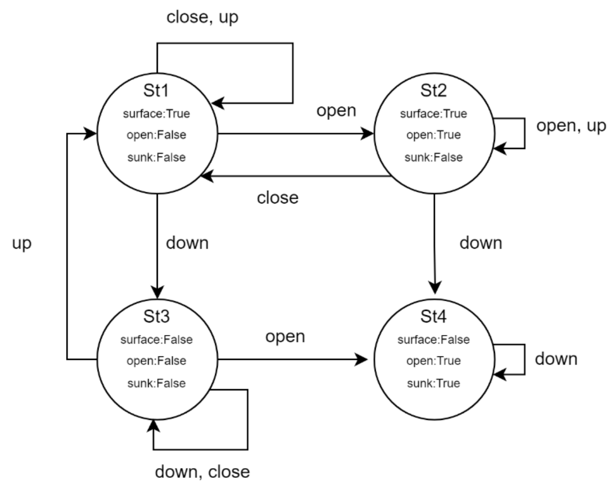

Figure 1: Graphical submarine model

2.2 CTL Expression and Analysis

2.2.1 CTL definition

Express the property *"on all paths, always, the submarine is not sunk"* in CTL:

$$\mathrm{AG}\ (\neg \mathrm{sunk})$$

This means that for all paths λ starting from s_1, we have $\forall i \geq 0, \mathrm{sunk}(\lambda[i]) = \mathrm{False}$.

2.2.2 True in s1

Is this property true in s1? Explain your answer with the reference to CTL truth definitions.

If we analyse this property in s1, we can say that is not true. We can demonstrate that by showing a case of path in which this property is not fulfilled (counterexample). // Consider the path:

$$s_1 \xrightarrow{\text{down}} s_3 \xrightarrow{\text{open}} s_4.$$

Starting in the initial state (s1), the submarine performs the action down. Then, the state turns out to be s3. The submarine performs the action open, transiting from s3 to s4. State 4 establishes that the submarine is not on the surface, the hatch open, and the submarine is sunk. Then, exists one path in which s1 is included that the submarine is sunk (EF(sunk)). Therefore, property is not true in s1.

2.3 Another CTL Expression and Analysis

Express the property: *"there exists a path where, in some future state, the submarine is not on the surface and not sunk, and until that state, it holds that the submarine was also not sunk"*:

$$\mathrm{EF}\big((\neg\mathrm{surface} \wedge \neg\mathrm{sunk}) \wedge (\neg\mathrm{sunk}\mathrm{U}(\neg\mathrm{surface} \wedge \neg\mathrm{sunk}))\big).$$

An example path satisfying this is:

$$s_1 \xrightarrow{\text{down}} s_3,$$

where $V(s_3) = \{\}$ so it meets $(\neg\mathrm{surface} \wedge \neg\mathrm{sunk})$ and along the path sunk were never true, satisfying $(\neg\mathrm{sunk}\mathrm{U}(\neg\mathrm{surface} \wedge \neg\mathrm{sunk}))$

2.4 Differences Between CTL and CTL* with Finite Paths

In CTL*, paths are assumed to be infinite. However, CTL* with finite traces allows terminal states with no successors. This distinction makes some formulas valid in CTL but invalid in CTL* with finite paths.

For example, *EG* True is always valid in CTL due to the infinite nature of paths, but it may fail in CTL* with finite traces if all paths terminate.

2.6 Translation and witness for formulas

We provide here the translation of the statements to the CMAS syntax (CTL with slightly different symbols).

- AF surface and !open and !sunk: On all paths there is a future state where surface is true and open and sunk are false (this just describes the initial state, since it is included in all paths starting from it)

- AG surface → EF !surface: On all paths if surface is true then exists a path in which we can turn it false.
- AG !open → EF open: On all paths if open is false then exists a path in which we can turn it true.
- EF (EG (!surface and !sunk)): There is a path where from some future state, there is a path where in every state the submarine is not on the surface but is also not sunk.

The formula $EF(EG(\neg\text{surface} \wedge \neg\text{sunk}))$ is **true** in s1 given our model. A valid witness is:

$$\text{Path: } s_1 \xrightarrow{\text{down}} s_3 \to s_3 \to s_3 \ldots$$

As given by CMAS:

```
   Formula number 4: (EF (EG ((! surface) && (! sunk)))), is TRUE in the model
  The following is a witness for the formula:
   < 0 1 >
   < 1 1 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Submarine
  open = false
  sunk = false
  surface = true
----------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Submarine
  open = false
  sunk = false
  surface = false
----------------------------------------
```

State s_3 satisfies $\neg\text{surface} \wedge \neg\text{sunk}$, and looping within s_3 indefinitely makes the trace fulfill the formula.

2.7 Translation and counter example for formula

Again, here we will provide the translation of the statements to the CMAS syntax.

- AG !open: On all paths globally the hatch is never open.
- AF sunk: On all paths, at some point in the future the submarine is sunk.
- EF (sunk and EF (!sunk)): There is a path where at some point the submarine is sunk and from there there is a path where at some point in the future it is not sunk.
- AX (AX !sunk): On all paths in the next state it holds that on all paths in the next state the submarine is not sunk.

Again, as given by CMAS:

```
 The following is a counterexample for the formula:
   < 0 1 >
   < 1 2 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Submarine
  open = false
  sunk = false
  surface = true
----------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Submarine
  open = true
  sunk = false
  surface = true
----------------------------------------
------------- State: 2 -----------------
Agent Environment
Agent Submarine
  open = true
  sunk = true
```

```
  surface = false
-----------------------------------------
```

In this case the system just sinks, as there is no way to make it out of it afterwards ($AX\neg sunk$). Thus, there is no path that satisfies the formula.

2.8 Model checking algorithm for new CTL definition

The algorithm would go as follows:

$$\begin{array}{l} \qquad \text{case } \varphi' = E(\varphi\, U^{+} \psi) : \\ Q_1 \leftarrow \emptyset; \quad Q_2 \leftarrow [\psi]_M \cap [\varphi]_M \\ \text{while } Q_2 \not\subseteq Q_1 \text{ do} \\ \quad Q_1 \leftarrow Q_1 \cup Q_2 \\ \quad Q_2 \leftarrow \text{pre}_{\exists}(Q_1) \cap [\varphi]_M \\ [\varphi']_M \leftarrow Q_1 \end{array}$$

The final set Q_1 contains all states where there is a path where φ is true in every state, including the state where ψ becomes true.