Logics for Safe AI 2024/2025

Coursework 1

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

2.1 $F\phi$ equivalent to $FF\phi$

$F\phi \to FF\phi$

If $\lambda \models F\phi$, then $\lambda[i \ldots \infty) \models \phi$ for some $i \geq 0$. Thus, we know that for some $0 \leq j \leq i$, $\lambda[j \ldots \infty) \models F\phi$. Hence, for all $k \in [j, i]$, $\lambda[k \ldots \infty) \models F\phi$. This implies that $\lambda[i \ldots \infty) \models F\phi$, which occurs if and only if $\lambda \models FF\phi$.

$FF\phi \to F\phi$

If $\lambda \models FF\phi$, then $\lambda[i \ldots \infty) \models F\phi$ for some $i \geq 0$. Hence, we know that for some $j \geq i$, $\lambda[j \ldots \infty) \models \phi$. Therefore, $\lambda \models F\phi$.

2.2 $G\phi$ equivalent to $GG\phi$

$G\phi \to GG\phi$

If $\lambda \models G\phi$, then $\lambda[i \ldots \infty) \models \phi$ for any $i \geq 0$. Thus, it also holds for any $\lambda[j \ldots \infty)$ given $j \geq i$. Therefore, $\lambda[i \ldots \infty) \models G\phi$, which is true if and only if $\lambda \models GG\phi$, satisfying the condition.

$GG\phi \to G\phi$

If $\lambda \models GG\phi$, then $\lambda[i \ldots \infty) \models G\phi$ for any $i \geq 0$. Hence, for all $j \geq i$, $\lambda[j \ldots \infty) \models \phi$. Therefore, $\lambda[i \ldots \infty) \models \phi$, which is the definition of $\lambda \models G\phi$, satisfying the condition again.

2.2 $\phi \mathcal{M} \psi$ is equivalent to $\psi \mathcal{U}(\phi \wedge \psi)$

Proof:

$\phi\mathcal{M}\psi \to \psi\mathcal{U}(\phi\wedge\psi)$: $\lambda \models \phi\mathcal{M}\psi$ if and only if $\lambda[i \ldots \infty) \models \phi$ for some $i \geq 0$ and $\lambda[j \ldots \infty) \models \psi$ for all $0 \leq j \leq i$. Therefore, if $\lambda[j \ldots \infty) \models \psi$ for all j until i, then $\lambda[i \ldots \infty) \models \phi \wedge \psi$, which means that $\lambda \models \psi\mathcal{U}(\phi \wedge \psi)$.

$\psi\mathcal{U}(\phi\wedge\psi) \to \phi\mathcal{M}\psi$: $\lambda \models \psi\mathcal{U}(\phi\wedge\psi)$ if and only if $\lambda[j \ldots \infty) \models (\phi\wedge\psi)$ for some $j \geq 0$ and $\lambda[i \ldots \infty) \models \psi$ for all $j \geq i \geq 0$. Since $\lambda[j \ldots \infty) \models (\phi \wedge \psi)$, it follows that $\lambda[i \ldots \infty) \models \phi$. Thus, $\lambda \models \phi\mathcal{M}\psi$.

2.3 Path from state 3 that falsifies Fp_0:

Example: $3 \to 5 \to 4 \to 3 \to 5 \to 4 \to \ldots$ looping on the cycle $3, 5, 4$.

All paths from state 2 satisfy $(GFp_0) \to (G(p_0 \to XXp_4))$:

All paths must start with $2 \to 4 \to 3 \to 5$ and then can loop arbitrarily in either $5, 4, 3$ or $5, 6, 3$. If they remain in the loop, they do not support (GFp_0), so the implication is not checked. After that, they can escape via $6 \to 1$ and either loop in $1, 0, 2, 4, 3, 5, 6$ or avoid 0. For (GFp_0) to hold, all paths must eventually loop and visit 0. The only way to get there is through $0 \to 2 \to 4$, as there are no other valid paths to continue the loop.

2.4 Path satisfying $Gp \wedge Fq$ but not $G(p \wedge Fq)$:

A path $s_0, s_1, s_2, \ldots$ satisfies $Gp \wedge Fq$ but does not satisfy $G(p \wedge Fq)$ if p is true in all states (Gp), q is true at some future moment (Fq), and from some state s_i, Fq is no longer true.

Example:

Index i of state s_i	p	q
0	true	false
1	true	true
2, ...	true	false

In this example: - p is true in all states (Gp). - q is true in state s_1, satisfying Fq. - After state s_1, q is no longer true, so $G(p \wedge Fq)$ is not satisfied.

3.1 Constructing a Shield from the Safety Game

Suppose while constructing a shield, we end up with the safety game shown in Figure 1. The agent has two actions, “a” and “b” (in some states, only one of these actions is available), whereas the environment has two labels, “1” and “2”. All states except s_1 and s_3 are safe states.

The shield S would be defined as:

$$S = (G(\text{properties}), k),$$

where k represents the set of actions for each state such that any label given by the system ensures the state does not leave the winning set (Win set).

Computing the Winning Set

To compute the shield, we calculate the winning set of states. The winning set starts as the set of safe states $\{S_0, S_2, S_4, S_5\}$ and is refined iteratively by eliminating states where no action ensures remaining in the winning region. This process continues until the set does not change anymore.

Steps: 1. Initial Winning Set:

$$\text{Win}_0 = \{S_0, S_2, S_4, S_5\}$$

As there is no action for S_5 that ensures remaining in Win_0, we remove it.

2. Updated Winning Set:

$$\text{Win}_1 = \{S_0, S_2, S_4\}$$

Now, the action ”a” for S_4 (action ”b” is unavailable due to G) becomes insecure as it cannot ensure staying in Win_1. Thus, we remove S_4.

3. Updated Winning Set:

$$\text{Win}_2 = \{S_0, S_2\}$$

At this stage, there is no way for an automaton in S_2 to remain in Win_2. Hence, S_2 is removed.

4. Updated Winning Set:

$$\text{Win}_3 = \{S_0\}$$

For S_0, both actions ”a” and ”b” result in leaving Win_3. Thus, the winning set becomes empty:

$$\text{Win}_4 = \emptyset$$

This implies that $k(s)$ for all $s \in G$ is:

$$k(s) = \emptyset.$$

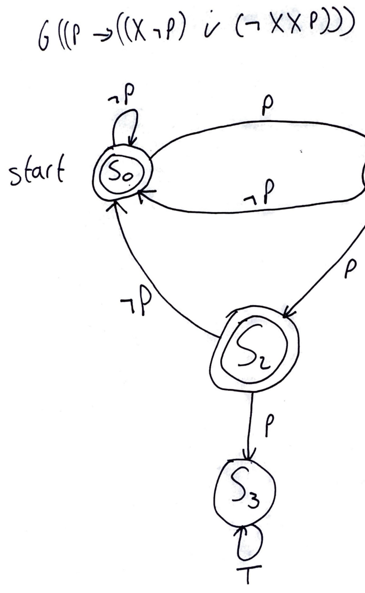

Figure 1: Automaton for exercise 3.3

3.2 Path Satisfying $Hq \to Pp$ but not $H(q \to Pp)$

A path s_0, s_1, s_2 satisfies $Hq \to Pp$ but not $H(q \to Pp)$ if q is true (Hq) and p must have been true for some states in the past (Pp). However, $q \to Pp$ does not hold historically ($H(q \to Pp)$) at every state.

Example

Index i of state s_i	q	p
0	false	false
1	true	false
2	true	true

In this example: - q is true historically (Hq), - p is true at some past states (Pp), - $q \to Pp$ does not hold historically for all states, violating $H(q \to Pp)$.

3.3 Automaton Conditions for Acceptance

For the automaton, the sequence is accepted only if, for each occurrence of p: 1. It is followed by $\neg p$ (solves the implication), or 2. It is followed by p and then $\neg p$ (solves the implication for both the last p and the new p).

If neither condition is satisfied, the automaton transitions to a state that is not accepted, as the condition for the first p is not solved.

3.4 Constraining Action "a" as in Figure 2

To be able to constrain action "a" as shown in Figure 2, we must take into consideration the following constraints:

- Action "a" is only allowed in state s_0.
- Action "a" is not allowed in states s_1 or s_2.
- Action "b" is allowed in any state.

These constraints can be expressed using the PPLTL formula:

$$a = Y(b, \neg p).$$

This formula states that action "a" is valid if and only if the prior action was "b" and p was false. Considering that s_0 is the only state with outgoing arrows labeled "a," we deduce the following from the shield:

- After performing action "b," p is false (in state s_0).

Practical Assignment: PPLTL Formulas

For the practical part we had to adapt the safety constraints to the following PPLTL Formulas:

$$H(\text{level} < 100) \wedge H(\text{level} > 0) \wedge$$
$$H((\text{open} \wedge Y\text{close}) \rightarrow YY\text{close}) \wedge$$
$$H((\text{close} \wedge Y\text{open}) \rightarrow YY\text{open}) \wedge ((P(\text{level} < 10) \wedge \text{close}) \rightarrow \text{level} \geq 50)$$

And for point 3 we adapt the following formulas:

$$\varphi_{\text{close}} = (P(\text{level} < 10) \to \text{level} \geq 50) \wedge \text{level} \geq 7 \wedge (Y\text{open} \to YY\text{open})$$

$$\varphi_{\text{open}} = \text{level} \leq 91 \wedge (Y\text{close} \to YY\text{close})$$

$$\varphi_{\text{noop}} = (\text{level} \leq 91 \wedge \text{level} \geq 7) \vee (\text{level} > 91 \wedge Y\text{close}) \vee (\text{level} < 7 \wedge Y\text{open})$$