Katholieke
Universiteit
Leuven

**Department of
Computer Science**

MCS PROJECT PART 2: SAVING ARKHAM AGAIN

Alexander Tang - s0189509
Xueying Deng - r0601458

Contents

1 LTL/CTL

1.1 When it is a player turn it is always possible to eventually win

Interpreted as: *If it's a player turn, then there exists at least one path that eventually leads to a win state.*

- **CTL:** AG ({state = player_turn} ⇒ EF {state = won})
- Returns true.

1.2 When the game is lost, it stays lost forever

- **LTL:** G ({state = lost} ⇒ G {state = lost})
- Returns true.

1.3 It is possible that the game goes on forever, i.e. it never reaches the won or lost state

- **CTL:** EG {state $\notin$ over}
- Returns true.

1.4 The game stays in the playing state until either the win or lose condition holds

- **LTL:** G ({state $\in$ playing} ⇒ ({card(open_gates) $\geqslant$ open_to_lose $\vee$ card(closed_gates) $\geqslant$ closed_to_win} R {state $\in$ playing}))
- Returns true. The given statement only holds if the game hasn't been lost or won yet. Furthermore, it's possible that the win or lose condition is never reached. That's why **Release** must be used instead of **Until**. It is also worth noting that for at least one state in the path after the win/lose condition has been reached, that *state* remains an element of *playing*. The *state* only changes to won or lost when the *win_game*/*lose_game* events have been called. This will not cause a problem since the release condition will remain true for the remainder of the path.

1.5 The win and lose condition cannot hold at the same time

- **LTL:** ¬F {card(open_gates) $\geqslant$ open_to_lose $\wedge$ card(closed_gates) $\geqslant$ closed_to_win}
- Returns true.

1.6 The game is always in exactly one of the playing states until at least the end of a turn

- **LTL:** G ({state $\in$ playing} U {turn_end = TRUE})
- Returns true. Seems rather trivial since the game always starts as a player turn and the turn will definitely end. If it was implied that every playing turn should stay in the playing turn until the end of the turn, then the following check should be used:
  G ({state $\in$ playing} ⇒ G ({state $\in$ playing} U {turn_end = TRUE}))

1.7 It is always true that if you eventually win, you eventually have more closed gates than open gates

- **LTL:** G (F ({state = won} ⇒ {card(closed_gates) > card(open_gates)}))
- Returns true. True for this game world, because 3 open gates are needed to lose and 2 closed gates are needed to win. In order to have equal or more open gates when the win condition is met, the step before must have at least 3 open gates so that there will be at least 2 open gates when one of them closes. This cannot happen since 3 open gates would mean we lost. In a game world where there are 2 or more extra gates required for the lose condition relative to the win condition, the given statement will no longer hold (and thus return false).

1.8 If there is a monster in the game, it stays in the game until the investigator attacks

For the purpose of verification, there is only one monster in the monsterpool.

- **LTL:** $G\ (\{\text{monsters} \neq \varnothing\} \Rightarrow ([\text{attack}]\ R\ \{\text{monsters} \neq \varnothing\}))$
- Returns true. The set of monsters will become empty when attacking since only one may exist at any time.

1.9 To win you have to eventually close two gates

Interpreted as: *There does not exist a single path where eventually the amount of closed gates is not equal to 2 (or closed_to_win) and the state becomes won at the same time.*

- **CTL:** $\neg EF\ (\{\text{state} = \text{won}\} \wedge \{\text{card(closed\_gates)} \neq 2\})$
- Returns true.

2 Additional information

- Monsters don't have to be added every monster turn, the option just becomes available. It's possible to move without adding a monster while a gate is open.
- Some proof obligations turn red on my computer (Windows), while they turn out green on the computers in the lab. Since the proof obligations won't be graded, I paid no further attention to this detail. Note that the implemented game simulation runs as it should.