Practical patterns of specifications

We may require some of the following properties of real systems:

- It is impossible to get to a state where started holds, but ready does not hold: G¬(started ∧ ¬ready)
- The negation of this formula expresses that it is possible to get to such a state, but this is only so if interpreted on paths (π φ). We cannot assert such a possibility if interpreted on states (s φ) since we cannot express the existence of paths; for that interpretation, the negation of the formula above asserts that all paths will eventually get to such a state.
- For any state, if a request (of some resource) occurs, then it will eventually be acknowledged: G (requested → F acknowledged).
- A certain process is enabled infinitely often on every computation path: G F enabled.
- Whatever happens, a certain process will eventually be permanently deadlocked: F G deadlock.
- If the process is enabled infinitely often, then it runs infinitely often. G F enabled → G F running.
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor: G (floor2 ∧ directionup ∧ ButtonPressed5 → (directionup U floor5)) Here, our atomic descriptions are boolean expressions built from system variables, e.g., floor2.

There are some things which are not possible to say in LTL, however. One big class of such things are statements which assert the existence of a path, such as these ones:

- From any state it is possible to get to a restart state (i.e., there is a path from all states to a state satisfying restart).
- The lift can remain idle on the third floor with its doors closed (i.e., from the state in which it is on the third floor, there is a path along which it stays there).

LTL can’t express these because it cannot directly assert the existence of paths.