# Goals as Reward-Generating Programs Domain Specific Language

May 14, 2024

BODY

## 1  Modal Definitions in Linear Temporal Logic

### 1.1  Linear Temporal Logic definitions

We offer a mapping between the temporal sequence functions defined in **??** (**??**) and linear temporal logic (LTL) operators. As we were creating this DSL, we found that the syntax of the $\langle then \rangle$ operator felt more convenient than directly writing down LTL, but we hope the mapping helps reason about how we see our temporal operators functioning. LTL offers the following operators, using $\varphi$ and $\psi$ as the symbols (in our case, predicates). Assume the following formulas operate sequence of states $S_0, S_1, \cdots, S_n$:

- **Next**, $X\psi$: at the next timestep, $\psi$ will be true. If we are at timestep $i$, then $S_{i+1} \vdash \psi$

- **Finally**, $F\psi$: at some future timestep, $\psi$ will be true. If we are at timestep $i$, then $\exists j > i : S_j \vdash \psi$

- **Globally**, $G\psi$: from this timestep on, $\psi$ will be true. If we are at timestep $i$, then $\forall j : j \geq i : S_j \vdash \psi$

- **Until**, $\psi U \varphi$: $\psi$ will be true from the current timestep until a timestep at which $\varphi$ is true. If we are at timestep $i$, then $\exists j > i : \forall k : i \leq k < j : S_k \vdash \psi$, and $S_j \vdash \varphi$.

- **Strong release**, $\psi M \varphi$: the same as until, but demanding that both $\psi$ and $\varphi$ are true simultaneously: If we are at timestep $i$, then $\exists j > i : \forall k : i \leq k \leq j : S_k \vdash \psi$, and $S_j \vdash \varphi$.

  *Aside:* there's also a **weak until**, $\psi W \varphi$, which allows for the case where the second is never true, in which case the first must hold for the rest of the sequence. Formally, if we are at timestep $i$, *if* $\exists j > i : \forall k : i \leq k < j : S_k \vdash \psi$, and $S_j \vdash \varphi$, and otherwise, $\forall k \geq i : S_k \vdash \psi$. Similarly there's **release**, which is the similar variant of strong release. We're leaving those two as an aside since we don't know we'll need them.

### 1.2  Satisfying a $\langle then \rangle$ operator

Formally, to satisfy a preference using a $\langle then \rangle$ operator, we're looking to find a sub-sequence of $S_0, S_1, \cdots, S_n$ that satisfies the formula we translate to. We translate a $\langle then \rangle$ operator by translating the constituent sequence-functions ($\langle once \rangle$, $\langle hold \rangle$, $\langle while\text{-}hold \rangle$)[1] to LTL. Since the translation of each individual sequence function leaves the last operand empty, we append a 'true' ($\top$) as the final operand, since we don't care what happens in the state after the sequence is complete.

(once $\psi$) := $\psi X \cdots$

(hold $\psi$) := $\psi U \cdots$

(hold-while $\psi \ \alpha \ \beta \cdots \nu$) := $(\psi M \alpha) X (\psi M \beta) X \cdots X (\psi M \nu) X \psi U \cdots$ where the last $\psi U \cdots$ allows for additional states satisfying $\psi$ until the next modal is satisfied.

For example, a sequence such as the following, which signifies a throw attempt:

```
(then
    (once (agent_holds ?b))
    (hold (and (not (agent_holds ?b)) (in_motion ?b)))
    (once (not (in_motion ?b)))
)
```

Can be translated to LTL using $\psi$ := (agent_holds ?b), $\varphi$ := (in_motion ?b) as:

$\psi X (\neg \psi \wedge \varphi) U (\neg \varphi) X \top$

Here's another example:

---

[1]These are the ones we've used so far in the interactive experiment dataset, even if we previously defined other ones, too.

```
(then
    (once (agent_holds ?b))    α
    (hold-while
        (and (not (agent_holds ?b)) (in_motion ?b))   β
        (touch ?b ?r)   γ
    )
    (once  (and (in ?h ?b) (not (in_motion ?b))))   δ
)
```

If we translate each predicate to the letter appearing in blue at the end of the line, this translates to:

$\alpha X(\beta M\gamma)X\beta U\delta X\top$