Planning and Reasoning

Gallotta Roberto

Wednesday 16th December, 2020

1 LTL

Modal TL that also contains two more fine-grained operators still referred to time. A **trace** is a sequence of propositional interpretations; it's a complete description of evolution over time. The modal operators are:

- G: Like $\Box$, means that something is true now and forever
- F: Like $\Diamond$, something will be true in the future or is true now (something true at least some point in the future)
- X: Something is true in the next time point
- U: A is true until B becomes true

Example:

Any no-modal formula evaluates on the first state: $x \vee \neg y$, for example, evaluates to true.

- $X\neg y$ is true because y is false in the second state
- XXy is false because y is false in the third state
- Gx is true because x is true in all states
- Gy is false because y is not true in all states
- GFy is true because Fy is true in any time point
- $xU\neg y$ is true: it doesn't matter how many xs we have, the important thing is that then we have $\neg y$
- $X(yU\neg x)$ is false: from the next state, y is already false and x is always true
- $yUXXy$ is true: we must have y true until it's false, then we check if XXy holds. In this trace, it's true because y is false in the second time point and XXy is true there (y is true in the fourth time point)
- $G(yUXXy)$ can't be checked directly

1.1 Semantics

A trace is a Kripke model. The difference is that in a trace every s_i is linked to every s_j for every $i \leq j$. In a general Kripke model there is no way to define what is the *next* state. A trace is a sequence of time points $[s_i, s_{i+1}, s_{i+2}, ...]$ and we always evaluate the formula in s_0. If we want to evaluate it further in the trace, we remove the previous time points and retain the rest (we don't need to keep the past).

We will say that a formula is true or false in a trace, not in a model. We use this also for non-modal formulas. For instance, x is true in the trace if x is true in s_0. $A \wedge B$ is true if A is true in the trace and B also is true in the trace. Same for the other formulas. For the modal formulas:

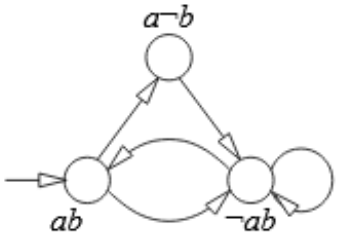
- XA is true if A is true in $[s_1, s_2, ...]$
- FA is true if there exists an i such that A is true in $[s_i, s_{i+1}, s_{i+2}, ...]$

- GA is true if, for all i, A is true in $[s_i, s_{i+1}, s_{i+2}, ...]$
- AUB is true if there exists an i such that:
  - B is true in $[s_i, s_{i+1}, s_{i+2}, ...]$
  - For all j such that $0 \leq j < i$, A is true in $[s_j, s_{j+1}, s_{j+2}, ...]$

1.2 Structure

A **structure** is the description of the evolution of some domain. It's a compact representation. The arrows represent different paths of evolution of the system. We denote with a short arrow the initial state of the trace.

Example:



From this structure we could have the following trace: $[ab, a\neg b, \neg ab, ab, a\neg b, ...]$, following the arrows. It's one possible evolution of the system over time. Another possible trace is $[ab, \neg ab, \neg ab, \neg ab, ...]$, looping in a state. Yet another possible trace is $[ab, a\neg b, ab, a\neg b, ab, a\neg b, ...]$. Another possible trace is $[ab, a\neg b, \neg ab, ab, \neg ab, \neg ab, \neg ab, ...]$, where we made a different choice the second time we got in a state.

1.2.1 Evaluation

When we have to move to the next state, we have to move both in time (to the next time point) but also we have to choose which action to take. So for a given modal formula we have to first choose the successor and then follow it to evaluate the formula.

Example using the previous structure: we want to check whether Xa is true (in ab). We have two arrows to choose from. One choice is to move to $a\neg b$, the other to $\neg ab$. Then, the formula is not true in both successors, so Xa is no true. However, Xa is true in one of the two. The first checks if it's always true (*universal*), the second checks whether it can be true (true at least once) (*existential*). We choose between these two accordingly to the application (in some cases we're interested if there's a possibility that it's true, in some other cases we're interested if it's always true).

Formally, given a formula and a structure:

- Existential evaluation: the formula is true in at least one trace (example: Xa, Gb)
- Universal evaluation: the formula is true in all traces (example: $G(a \vee b)$, $G(\neg a \vee X\neg a \vee XX\neg a)$)

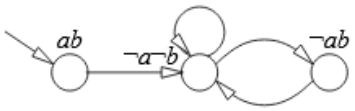
In LTL we can only check formulas in either existential way or universal way, no way to mix them. In CTL* we can ($AFx \vee EGx$, which means x, in all traces, is at some point true or x is true now and forever).

1.2.2 Model Checking

It's a way to evaluate a formula in a structure. We use the labelling algorithm. In order to check satisfiability of the formula we need to check the satisfiability of the component's formulas. We proceed from the subformulas to the formula.

Example:

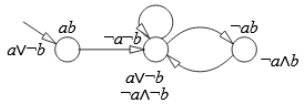
$$(a \vee \neg b)U(F(\neg a \wedge \neg b) \wedge X(\neg a \wedge b))$$



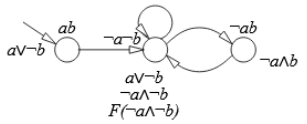
The first step is to recursively break the formula into its subformulas (disregarding duplicates):

1. $(a \vee \neg b)U(F(\neg a \wedge \neg b) \wedge X(\neg a \wedge b))$
2. $a \vee \neg b$, $F(\neg a \wedge \neg b) \wedge X(\neg a \wedge b)$
3. $F(\neg a \wedge \neg b)$, $X(\neg a \wedge b)$
4. $\neg a \wedge \neg b$, $\neg a \wedge b$

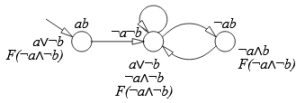
We only use these formulas in the algorithm since they're the only ones that are relevant to the original formula; we are working on a specific subset of formulas only (the truth of any other formula is irrelevant to our problem). Then we check where the subformula is true in a bottom-up order.



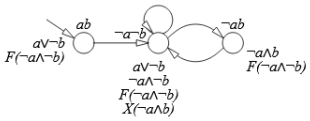
We start by labeling the nodes where the propositional formulas are true.



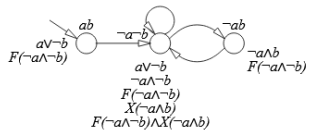
Then we can move on to the modal operators. We start where the subformula of the modal formula is true (we know this since we labeled the states). If $\neg a \wedge \neg b$, then $F(\neg a \wedge \neg b)$ is also true.



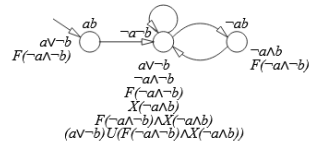
We do this from the current state to its predecessors recursively (since $F(\neg a \wedge \neg b)$ is true in the initial node).



For the X operator, we start from where it's true ($\neg a \wedge b$, the third node) and we mark the previous (second) node as true for $X(\neg a \wedge b)$. This formula is true if the subformula is true in the direct successor.



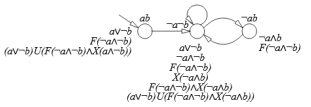
Then we move to the conjunction. Following the semantics of LTL, we mark only the nodes where both $F(\neg a \wedge \neg b)$ and $X(\neg a \wedge b)$ are true.



Now for the U modal formula. If we have a state where a subformula of U is true, then we also have U true (if A is true, then BUA is true).

Then we mark U recursively: we mark the states where B is true and the BUA is true in the next state. We can formalize this as:

- Preconditions:
  1. The successor has the entire U formula
  2. The predecessor is linked to its successor
  3. The predecessor satisfies the B part of BUA formula
- Consequence:
  1. The BUA formula is true in the predecessor



If the original formula is in the initial state, the formula is true in the structure.