Course Script

IN 5110: Specification and Verification of Parallel Systems

IN5110, autumn 2019

Martin Steffen, Volker Stolz

Contents

Chapter 1
LTL model checking

What is it about?

Learning Targets of this Chapter

The chapter covers LTL and how to do model checking for that logic, using Büchi-automata.

Contents

1.1 Introduction

In this chapter, we leave behind a but the "logical" treatment of logics (like asking for validity etc, i.e., asking $\models \varphi$), but proceed to the question of *model checking*, i.e., when does a concrete model satisfies a formula $M \models \varphi$. We do that for a specific modal logic, more precisely temporal logic. It's one of the most prominent and the first one that taken up seriously in computer science (as opposed to mathematics or philosphy). We will also cover *one* central way of doing model checking of temporal logics, namely *automata-based* model checking.

Temporal logic?

LTL

1.2 LTL

LTL: speaking about "time"

In **Linear Temporal Logic (LTL)**, also called linear-time temporal logic, we can describe such properties as, for instance, the following: assume time is a *sequence* of discrete points i in time, then: if i is *now*,

- p holds in i and every following point (the future)
- p holds in i and every preceding point (the past)

$$\ldots \longrightarrow \bullet^{p}_{i-2} \longrightarrow \bullet^{p}_{i-1} \longrightarrow \bullet^{p}_{i} \longrightarrow \bullet^{p}_{i+1} \longrightarrow \bullet^{p}_{i+2} \longrightarrow \ldots$$

Time here is *linear* and *discrete*. One consequently just uses ordinary natural numbers (or integers) to index the points in time. We will mostly only be concerned with the future, i.e., we won't good much into past-time LTL resp. versions of LTL that allow to speak about the future *and* the past. Branching time is an alternative to the linear modelling of time, and instead of having discrete point in times, one could have dense time and/or deal with intervals.

1.2.1 Syntax

Syntax

As before, we start with the syntax of the logic at hand, given by a grammar. We assume some underlying "core" logic, like *propositional* logic or *first-order* logic. Focusing on the temporal part of the logic, we don't care much about that underlying core. Practically, when it comes to automatically checking, the choice of the underlying logic of course has an impact. But we treat the handling of the underlying logic as *orthogonal*. The first thing to extend is the syntax: we have formulas ψ of said underlying core, and then we extend it but the temporal operators of LTL, adding $\Box$, $\Diamond$, $\bigcirc$, U, R, and W. So the syntax of (a version of) LTL is given by the following grammar.

ψ			propositional/first-order formula
φ	$::=$	ψ	formulas of the "core" logics
	$\mid$	$\neg\varphi \mid \varphi \wedge \varphi \mid \varphi \to \varphi \mid \ldots$	boolean combinations
	$\mid$	$\bigcirc\varphi$	next φ
	$\mid$	$\Box\varphi$	always φ
	$\mid$	$\Diamond\varphi$	eventually φ
	$\mid$	$\varphi\ U\ \varphi$	"until"
	$\mid$	$\varphi\ R\ \varphi$	"release"
	$\mid$	$\varphi\ W\ \varphi$	"waiting for", "weak until"

As in earlier logics, one can ponder, whether the syntax is *minimal*, i.e., do we need all the operators, or can some be expressed as syntactic sugar by using others? The answer is: the syntax is *not minimal*, some operators can be left out and we will see that later. For a robust answer to the question of minimality, we need to wait until we have clarified the meaning, i.e., until we have defined the semantics of the operators.

1.2.2 Semantics

Fixing the meaning of LTL formulas means, to define a semantical satisaction relation $\models$ between "models" and LTL formulas. In principle, we know how that could work, having seen similar definitions when discussing modal logics in general (using Kripke frames, valuations, and Kripke models). Now, that we are dealing with a *linear* temporal logic, the Kripke frames should be also of linear structure. What kind of *valuations* we employ would depend on the underlying logics. For example for propositional LTL, one needs an interpretation of the propositional atoms per world, for first-order LTL, one needs a

choice of the free variables in the terms and formulas (the signature and its interpretation does not change when going from one world to another, only, potentially, the values of the variables).

That's also what we do next, except that we won't use explicitly the terminology of *Kripke frame* or Kripke model. We simply assume a sequence of discrete time points, indexed by natural numbers. So the numbers i, $i+1$ etc. denote the worlds, and the accessibility relation simply connects a "world" i with its successor world $i+1$. As was done with Kripke models, we then need a valuation per world, i.e., per time point. In the case of propositional LTL, it's a mapping from propositional variables to the boolean values $\mathbb{B}$. To be consistent with common terminology, we call such a function $P \to \mathbb{B}$ here not a valuation, but a *state* (but see also the side remarks about terminology below). Let's use the symbol s to represent such a state or valuation. A *model* then provides a state per world, i.e., a mapping $\mathbb{N} \to (P \to \mathbb{B})$. This is equivalently represented as an infinite sequence of the form

$$s_0 s_1 s_2 \dots$$

where s_0 represents the state at the zero'th position in the infinite sequence, s_1 at the position or world one after that etc. Such an infitite sequence of states is called *path*, and we use letters π, π' etc. to refer to them them. It's important to remember that paths are *infinite*. As discussed in the lecture: if we allowed *finite* paths, we would loose the kind of nice duality between the $\Diamond$ and $\Box$ operator (that refers to the fact that that $\neg\Box\neg$ is the same as $\Diamond$, and the other way around).

In that connection: what's $\neg \bigcirc \neg$?

Some remarks on terminology: paths, states, and valuations

The notions of states and paths ... are slightly differring in the literature. It's not a big problem as the used terminology is not incompatible, just sometimes not in complete agreement.

For example, there is a notion of path in connection with graphs. Typically, a path in a graph from a node n_1 to a node n_2 is a sequence of nodes that follows the edges of the given graph and that starts at n_1 and ends in n_2. The length of the path is the number of *edges* (and with this definition, the *empty* paths from n to n contains one node, namely n). There maybe alternative definitions of paths in "graph theory" (like sequences of edges). In connection with our current notion of paths, there are 2 major differences. Our paths are *ininite*, whereas when dealing with graphs, a path normally is understood as a *finite sequence*. There is no fundamental reason for not considering (also) infinite paths there (and some people surely do), it's just that the standard case there is finite sequence, and therefore the word *path* is reserved for those. LTL on the other hand deals with ininite sequences, and consequently uses the word paths for that.

The other difference is that a path here is not defined as "a sequence of nodes connected *by edges*". It's simply an infinite sequence of valuations (and the connection is just by the position in the sequence), there is no question of "is there a transition from state at place i to that of at place $i+1$. Later, when we connect the current notion of paths to "path

through a transition system", then the more states in that infinite sequence need to arise by connecting transistions or edges in the underlying transition system or graph.

Finally, of course, the conventional not of path in a graph does not speaks of valuations, it's just a sequence of nodes. If N is the set of nodes of a graph, and $\mathbb{N}_n$ the finite set $\{i \in \mathbb{N} \mid i < n\}$, then a traditional path (of length n) in graphs is a function $\mathbb{N}_n \to N$ such that it "follows the edges".

There are other names as well, when it comes to linear sequences of "statuses" when running a program. Those include *runs*, *executions* (also traces, logs, histories etc). Sometimes they correspond to sequences of edges (for instance containing transition labels only. Sometimes they corrspond to sequences of "nodes" (containing "status-related" information like here), sometimes both.

Paths and computations

Definition 1.2.1 (Path)**.**

- A *path* is an infinite sequence

$$\pi = s_0, s_1, s_2, \dots$$

  of states.
- π^k denotes the *path* $s_k, s_{k+1}, s_{k+2}, \dots$
- π_k denotes the *state* s_k.

It's intended (later) that paths represent behavior of programs resp. "going" through a Kripke-model or transition system. A transitions system is a graph-like structure (and may contain cycles), and a paths can be generated following the graph structure. In that sense it corresponds to the notion of *paths* as known from graphs (remember that the mathematical notions of graph corresponds to Kripke frames). Note, however, that we have defined *path* independent from an underlying program or transition system. It's not a "path through a transition system", but it's simply an infinite sequence of state (maybe caused by a transition system or maybe also not).

Now, what's a *state* then? It depends on what kind of LTL we are doing, basically propositional LTL or first-order LTL. A state basically is the interpretation of the underlying logic in the given "world", i.e., the given point in time (where time is the index inside the linear path). In propositional logic, the state is the interpretation of the propositional symbols (or the set of propositional symbols that are considered to be true at that point). For first order logic, it's a valuation of the free variables at that point. When one thinks of modelling programs, then that's corresponds to the standard view that the state of an imperative program is the value of all its variables (= state of the memory).

1.2.3 Semantics

The satisfaction relation $\pi \models \varphi$ is defined inductively over the structure of the formula. We assume that for the formulas of the "underlying" core logic, we have an adequate $\models_{\mathsf{ul}}$ available, that works on *states*. Note that in case of first-order logic, a signature and its *interpretation* is assumed to be fixed.

Definition 1.2.2 (Satisfaction)**.** An LTL formula φ is *true* relative to a path π, written $\pi \models \varphi$, under the following conditions:

$$
\begin{array}{lll}
\pi \models \psi & \text{iff} & \pi_0 \models_{\mathsf{ul}} \psi \text{ where } \psi \text{ in underlying core language} \\
\pi \models \neg\varphi & \text{iff} & \pi \not\models \varphi \\
\pi \models \varphi_1 \wedge \varphi_2 & \text{iff} & \pi \models \varphi_1 \text{ and } \pi \models \varphi_2 \\
\pi \models \bigcirc\varphi & \text{iff} & \pi^1 \models \varphi \\
\pi \models \varphi_1 \ U \ \varphi_2 & \text{iff} & \pi^k \models \varphi_2 \text{ for some } k \geq 0, \text{ and} \\
 & & \pi^i \models \varphi_1 \text{ for every } i \text{ such that } 0 \leq i < k
\end{array}
$$

The definition of $\models$ covered $\bigcirc$ and U as the only temporal operators. It will turn out that these two operators are "complete" insofar that one can express remaining operators from the syntax by them. Those other operators are $\Box$, $\Diamond$, R, and W, according to the syntax we presented earlier. That's a common selection of operators for LTL, but there are sometimes even more added for the sake of convenience and to capture commonly encountered properties a user may wish to express.

We could explain those missing operators as syntactic sugar, showing how they can be macro-exanded into the core operators. What we (additionally) do first is giving a *direct* semantic definition of their satisfaction. As mentioned already earlier, the two important temporal operators "always" and "eventually" are written symbolically like the modal operators necessity and possibility, namely as $\Box$ and $\Diamond$, but their interpretation is slightly different from them. Their semantic definition is straightforward, referring to *all* resp. for *some* future point in time.

The release operator is the dual to the until operator, but is also a kind of "until" only with the roles of the two formulas exchanged. Intuitively, in a formula $\varphi_1 \ R \ \varphi_2$, the φ_1 "releases" φ_2's need to hold, i.e., φ_2 has to hold up until and *including* the point where φ_1 first holds and if φ_1 never holds (i.e., never "realeases φ_2), then φ_2 has to hold forever. If there a point where φ_1 is first true and thus releases φ_2, then at that "release point" both φ_1 and φ_2 have to hold. Furthermore, it's a "weak" form of a "reverse until" insofar that it's not required that φ_1 ever releases φ_2.

$$\pi \models \Box\varphi \qquad \text{iff } \pi^k \models \varphi \text{ for all } k \geq 0$$
$$\pi \models \Diamond\varphi \qquad \text{iff } \pi^k \models \varphi \text{ for some } k \geq 0$$

$$\pi \models \varphi_1 \; R \; \varphi_2 \quad \text{iff for every } j \geq 0, \text{ if } \pi^i \not\models \varphi_1 \text{ for every } i < j \text{ then } \pi^j \models \varphi_2$$

$$\pi \models \varphi_1 \; W \; \varphi_2 \text{ iff } \pi \models \varphi_1 \; U \; \varphi_2 \text{ or } \pi \models \Box\varphi_1$$

Validity and semantic equivalence

Definition 1.2.3 (Validity and equivalent)**.**

- φ is *(temporally) valid*, written $\models \varphi$, if $\pi \models \varphi$ for all paths π.
- φ_1 and φ_2 are *equivalent*, written $\varphi_1 \sim \varphi_2$, if $\models \varphi_1 \leftrightarrow \varphi_2$ (i.e. $\pi \models \varphi_1$ iff $\pi \models \varphi_2$, for all π).

Example 1.2.4. $\Box$ distributes over $\wedge$, while $\Diamond$ distributes over $\vee$.

$$\Box(\varphi \wedge \psi) \sim (\Box\varphi \wedge \Box\psi)$$
$$\Diamond(\varphi \vee \psi) \sim (\Diamond\varphi \vee \Diamond\psi)$$

Now that we know the semantics, we can transport other semantical notions to the setting of LTL. Validity, as usual captures "unconditional truth-ness" of a formula. In this case, it means thus, a formula that holds for all paths.

In a way, especially from the perspective of model checking, valid formulas are "boring". They express some universal truth, which may be interesting and gives insight to the logics. But a valud formula is also *trivial* in the technical sense in that it does not express any interesting properties. After all, it's equivalent to the formula $\top$. In other words, it's equallly useless as a specification as a contradictory formula (one that is equivalent to $\bot$), as it holds for all systems, no matter what.

Valid formulas may still be useful. If one knows that one property implies another (resp. that $\varphi_1 \to \varphi_2$ is valid), one could model-check using formula φ_1 (which might be easier), and use that to establish that also φ_2 holds for a given model. But still, unlike in logic and theorem proving, the focus in model checking is not so much on finding methods to derive or infer valid formulas.

However, the two problems — $M \models \varphi$ vs. $\models \varphi_1 \to \varphi_2$ — are not

The next illustrations are for propositional LTL, where we use p, q and similar for propositional atoms. We also indicate the states by "labelling" the corresponding places in the infinite sequence by mentioning the propositional atoms which are assume to hold at that point (and not mentioning those which are not). However, those are illustrations. For instance, when illustrating $\pi \models \bigcirc p$

1.2.4 Illustration

$\pi \models \Box p$

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^p \longrightarrow \bullet_4^p \longrightarrow \dots$$

$\pi \models \Diamond p$

$$\bullet_0 \longrightarrow \bullet_1 \longrightarrow \bullet_2 \longrightarrow \bullet_3^p \longrightarrow \bullet_4 \longrightarrow \dots$$

$\pi \models \bigcirc p$

$$\bullet_0 \longrightarrow \bullet_1^p \longrightarrow \bullet_2 \longrightarrow \bullet_3 \longrightarrow \bullet_4 \longrightarrow \dots$$

1.2.5 Some more illustrations

$\pi \models p \ U \ q$ (sequence of p's is *finite*)

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^q \longrightarrow \bullet_4 \longrightarrow \dots$$

$\pi \models p \ R \ q$ (sequence of qs may be infinite)

$$\bullet_0^q \longrightarrow \bullet_1^q \longrightarrow \bullet_2^q \longrightarrow \bullet_3^{p,q} \longrightarrow \bullet_4 \longrightarrow \dots$$

$\pi \models p \ W \ q$

The sequence of ps may be infinite. ($p \ W \ q \sim p \ U \ q \vee \Box p$).

$$\bullet_0^p \longrightarrow \bullet_1^p \longrightarrow \bullet_2^p \longrightarrow \bullet_3^p \longrightarrow \bullet_4^p \longrightarrow \dots$$

1.2.6 The Past

The past

Observation

- Manna and Pnueli [2] uses pairs (π, j) of paths and positions instead of just the path π because they have *past-formulae*: formulae without future operators (the ones we use) but possibly with *past operators*, like $\Box^{-1}$ and $\Diamond^{-1}$.

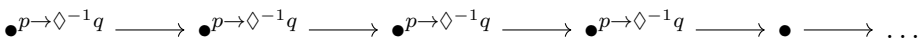
$$(\pi, j) \models \Box^{-1}\varphi \quad \text{iff} \quad (\pi, k) \models \varphi \text{ for all } k,\ 0 \leq k \leq j$$
$$(\pi, j) \models \Diamond^{-1}\varphi \quad \text{iff} \quad (\pi, k) \models \varphi \text{ for some } k,\ 0 \leq k \leq j$$

- However, it can be shown that for any formula φ, there is a *future-formula* (formulae without past operators) ψ such that

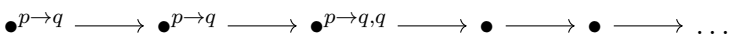
$$(\pi, 0) \models \varphi \quad \text{iff} \quad (\pi, 0) \models \psi$$

The past: example

$\Box(p \to \Diamond^{-1} q)$**?** $\quad (\pi, 0) \models \Box(p \to \Diamond^{-1} q)$



$(\pi, 0) \models q \ R \ (p \to q)$



1.2.7 Examples

Some examples

Temporal properties

1. If φ holds initially, then ψ holds eventually.
2. Every φ-position is responded by a later ψ-position (**response**)
3. There are **infinitely** many ψ-positions.
4. Sooner or later, φ will hold *permanently* (**permanence**, **stabilization**).
5. The first φ-position must coincide or be preceded by a ψ-position.
6. Every φ-position initiates a sequence of ψ-positions, and if terminated, by a χ-position.

Formalization of "informal" properties

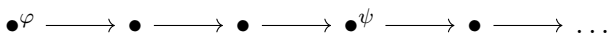
It can be difficult to correctly formalize informally stated requirements in temporal logic.

Informal statement: "φ implies ψ"

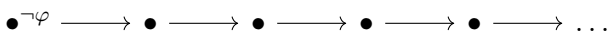
- $\varphi \to \psi$? $\varphi \to \psi$ holds in the initial state.
- $\Box(\varphi \to \psi)$? $\varphi \to \psi$ holds in every state.
- $\varphi \to \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.
- $\Box(\varphi \to \Diamond\psi)$? "response"

It is not obvious, which one of them (if any) is necessarily what was intended.

Example 1.2.5. $\varphi \to \Diamond\psi$: If φ holds initially, then ψ holds eventually.



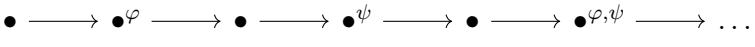
This formula will also hold in every path where φ does not hold initially.



Response

Example 1.2.6 (Response). $\Box(\varphi \to \Diamond\psi)$

Every φ-position coincides with or is followed by a ψ-position.

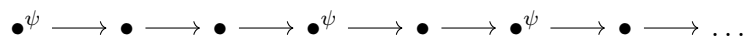


This formula will also hold in every path where φ never holds.

Example: ∞

Example 1.2.7. $\Box\Diamond\psi$

There are infinitely many ψ-positions.

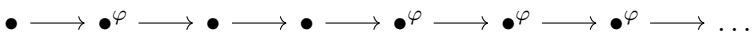


- model-checking?
- run-time verification?

Note that this formula can be obtained from the previous one, $\Box(\varphi \to \Diamond\psi)$, by letting $\varphi = \top$: $\Box(\top \to \Diamond\psi)$.

Permanence

$\Diamond\Box\varphi$ Eventually φ will hold *permanently.*

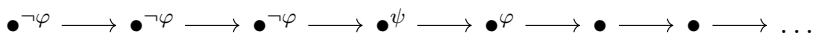


Equivalently: there are *finitely* many $\neg\varphi$-positions.

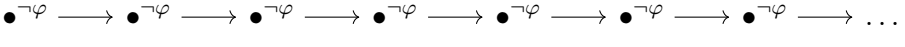
And another one

Example 1.2.8. $(\neg\varphi)\ W\ \psi$

The first φ-position must coincide or be preceded by a ψ-position.



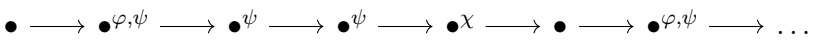
φ may never hold



LTL example

Example 1.2.9. $\square(\varphi \to \psi \ W \ \chi)$

Every φ-position initiates a sequence of ψ-positions, and if terminated, by a χ-position.



The sequence of ψ-positions need not terminate.

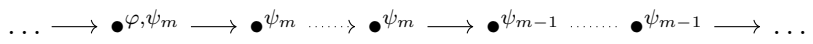
Nested waiting-for

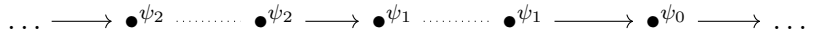
A **nested waiting-for formula** is of the form

$$\square(\varphi \to (\psi_m \ W \ (\psi_{m-1} \ W \cdots (\psi_1 \ W \ \psi_0) \cdots))),$$

where $\varphi, \psi_0, \ldots, \psi_m$ in the underlying logic. For convenience, we write

$$\square(\varphi \to \psi_m \, W \, \psi_{m-1} \, W \, \cdots \, W \, \psi_1 \, W \, \psi_0).$$





Explanation Every φ-position initiates a succession of intervals, beginning with a ψ_m-interval, ending with a ψ_1-interval and possibly terminated by a ψ_0-position. Each interval may be empty or extend to infinity.

Duality

Definition 1.2.10 (Duals)**.** For binary boolean connectives $\circ$ and $\bullet$, we say that $\bullet$ is the *dual* of $\circ$ if

$$\neg(\varphi \circ \psi) \sim (\neg\varphi \bullet \neg\psi).$$

Similarly for unary connectives: $\bullet$ is the dual of $\circ$ if $\neg \circ \varphi \sim \bullet\neg\varphi$.

Duality is *symmetric*:

- If $\bullet$ is the dual of $\circ$ then
- $\circ$ is the dual of $\bullet$, thus
- we may refer to two connectives as dual (of each other).

The $\circ$ and $\bullet$ operators are not concrete connectives or operators, they are meant as "placeholders". One can have a corresponding notion of duality for the unary operators $\Diamond$ and $\Box$, and even for null-ary "operators".

Dual connectives

- $\wedge$ and $\vee$ are duals:
$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$
- $\neg$ is its own dual:
$$\neg\neg\varphi \sim \neg\neg\varphi.$$
- What is the dual of $\rightarrow$? It's $\not\leftarrow$:
$$\begin{aligned}\neg(\varphi \not\leftarrow \psi) &\sim \varphi \leftarrow \psi \\ &\sim \psi \rightarrow \varphi \\ &\sim \neg\varphi \rightarrow \neg\psi\end{aligned}$$

Complete sets of connectives

- A set of connectives is *complete* (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., $\not\leftarrow$ can be defined), but also subsets of it, so we don't actually need all the connectives.

Example 1.2.11. <3-> $\{\vee, \neg\}$ is complete.

- $\wedge$ is the dual of $\vee$.
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \leftrightarrow \psi$ is equivalent to $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.
- $\top$ is equivalent to $p \vee \neg p$
- $\bot$ is equivalent to $p \wedge \neg p$

Duals in LTL

- What is the dual of $\Box$? And of $\Diamond$?
- $\Box$ and $\Diamond$ are duals.

$$\neg\Box\varphi \sim \Diamond\neg\varphi$$
$$\neg\Diamond\varphi \sim \Box\neg\varphi$$

- Any other?
- U and R are duals.

$$\neg(\varphi\ U\ \psi) \sim (\neg\varphi)\ R\ (\neg\psi)$$
$$\neg(\varphi\ R\ \psi) \sim (\neg\varphi)\ U\ (\neg\psi)$$

Complete set of LTL operators

Proposition 1. *The set of operators* $\vee, \neg, U, \bigcirc$ *is* complete *for LTL.*

We don't need all our temporal operators either.

Proof.
- $\Diamond\varphi \sim \top\ U\ \varphi$
- $\Box\varphi \sim \bot\ R\ \varphi$
- $\varphi\ R\ \psi \sim \neg(\neg\varphi\ U\ \neg\psi)$
- $\varphi\ W\ \psi \sim \Box\varphi \vee (\varphi\ U\ \psi)$

□

1.2.8 Classification of properties

We have seen a couple of examples of specific LTL formulas, i.e., specific properties. Specific "shapes" of formulas are particularly usefule or common, and they sometimes get specific names. If we take $\bigcirc$ and U as a complete core of LTL, then already the shape $\top\ U\ \varphi$ is so useful that it does not only deserve a special name, it even has a special syntax or symbol, namely $\Diamond$. We have encountered other examples before as well (like *permanence*) and in the following we will list some more. Another very important classification or characterization of LTL formulas is the distinction between *safety* and *liveness*. Actually, one should see it not so much as a characterization of LTL formulas, but of *properties* (of paths). LTL is a specific notation to describe properties of paths (where a property corresponds to a set of paths). Of course not all sets of paths are expressible in LTL (why not?). The situation is pretty analogous to that of regular expressions and regular languages. Regular expressions play the rule of the syntax and they are interpreted as sets of finite words, i.e., as *properties* of words. Of course not all properties of words, i.e. languages, are in fact regular, there are non-regular languages (context-free languages etc.).

Coming back to the LTL setting: it's better to see the distinction between safety and liveness as a qualification on path *properties* (= sets or languages of infinite sequences

of states), but of course, we then see which kind of LTL formulas are capturing a safety property or a liveness property.

Note (again) that "safety" or "liveness" is not property of a paths, it's a property of path properties, so to say. In other words, there will be no LTL formula expressing "safety" (it makes no sense), there are LTL formulas which correspond to a safety property, i.e., expresse a property that belongs to the set of all safety properties.

There is a kind of "duality" between safety and liveness in that safety is kind of like the "opposite" of liveness, but it's not that properties fall exactly into these to categories. There are properties (and thus LTL formulas) that are neither saferty properties nor lifeness properties.

Classification of properties

We can classify properties expressible in LTL. *Examples*:

invariant $\Box\varphi$

"liveness" $\Diamond\varphi$

obligation $\Box\varphi \vee \Diamond\psi$

recurrence $\Box\Diamond\varphi$

persistence $\Diamond\Box\varphi$

reactivity $\Box\Diamond\varphi \vee \Diamond\Box\psi$

- φ, ψ: non-temporal formulas

The *invariant* is a prominent example of a safety property. Each *invariant* property is also a *safety* property. Some people even use the words synonymously (earlier ediitions of the lecture), but according to the consensus or majority opinion, one should distinction the notions. See for instance the rather authoritative textbook Baier and Katoen [1]. It's however true that *invariants* are perhaps the most typical, easiest, and important form of safety properties and they also represent the essence of them. In particular, if one informally stipulates that safety corresponds to "never something bad happens", then that translates well to an invariant (namely the complete absence of the bad thing: "always not bad"). That characterization of safety is due to Lamport

Safety (slightly simplified)

- important basic class of properties
- relation to testing and run-time verification
- informally "nothing bad ever happens"

Definition 1.2.12 (Safety/invariant)**.**

- A *invariant* formula is of the form

$$\Box\varphi$$

for some first-order/prop. formula φ.

- A *conditional safety* formula is of the form

$$\varphi \to \Box\psi$$

  for (first-order) formulae φ and ψ.

Safety formulae express *invariance* of some **state property** φ: that φ holds in every state of the computation.

Safety property example

Mutex *Mutual exclusion* is a safety property. Let C_i denote that process P_i is executing in the critical section. Then

$$\Box\neg(C_1 \wedge C_2)$$

expresses that it should always be the case that not both P_1 and P_2 are executing in the critical section.

Observe: the negation of a safety formula is a liveness formula; the negation of the formula above is the liveness formula

$$\Diamond(C_1 \wedge C_2)$$

which expresses that eventually it *is* the case that both P_1 and P_2 are executing in the critical section.

Liveness properties (simplified)

Definition 1.2.13 (Liveness)**.**

- A *liveness* formula is of the form

$$\Diamond\varphi$$

  for some first-order formula φ.
- A *conditional liveness* formula is of the form

$$\varphi \to \Diamond\psi$$

  for first-order formulae φ and ψ.
-

Liveness formulae *guarantee* that some event φ eventually happens: that φ holds in at least one state of the computation.

Connection to Hoare logic

- *Partial correctness* is a safety property. Let P be a program and ψ the post condition.

$$\Box(terminated(P) \to \psi)$$

- In the case of *full partial correctness*, where there is a precondition φ, we get a *conditional safety* formula,

$$\varphi \to \Box(terminated(P) \to \psi),$$

which we can express as $\{ \varphi \} P \{ \psi \}$ in Hoare Logic.

Total correctness and liveness

- *Total correctness* is a liveness property. Let P be a program and ψ the post condition.

$$\Diamond(terminated(P) \wedge \psi)$$

- In the case of *full total correctness*, where there is a precondition φ, we get a *conditional liveness* formula,

$$\varphi \to \Diamond(terminated(P) \wedge \psi).$$

Duality of partial and total correctness

Partial and total correctness are dual.

Let

$$PC(\psi) \triangleq \Box(terminated \to \psi)$$
$$TC(\psi) \triangleq \Diamond(terminated \wedge \psi)$$

Then

$$\neg PC(\psi) \sim PC(\neg\psi)$$
$$\neg TC(\psi) \sim TC(\neg\psi)$$

Obligation

Definition 1.2.14 (Obligation)**.**

- A *simple obligation* formula is of the form

$$\Box\varphi \vee \Diamond\psi$$

for first-order formula φ and ψ.
- An equivalent form is

$$\Diamond\chi \to \Diamond\psi$$

which states that some state satisfies χ only if some state satisfies ψ.

Obligation (2)

Proposition 2. *Every safety and liveness formula is also an obligation formula.*

Proof. This is because of the following equivalences.

$$\Box\varphi \sim \Box\varphi \vee \Diamond\bot$$
$$\Diamond\varphi \sim \Box\bot \vee \Diamond\varphi$$

and the facts that $\models \neg\Box\bot$ and $\models \neg\Diamond\bot$. □

Recurrence and Persistence

Recurrence

Definition 1.2.15 (Recurrence)**.**

- A *recurrence* formula is of the form

$$\Box\Diamond\varphi$$

  for some first-order formula φ.
- It states that infinitely many positions in the computation satisfies φ.

Observation

A response formula, of the form $\Box(\varphi \to \Diamond\psi)$, is equivalent to a recurrence formula, of the form $\Box\Diamond\chi$, if we allow χ to be a past-formula.

$$\Box(\varphi \to \Diamond\psi) \sim \Box\Diamond(\neg\varphi)\ W^{-1}\ \psi$$

Recurrence

Proposition 3. *Weak fairness*[1] *can be specified as the following recurrence formula.*

$$\Box\Diamond(enabled(\tau) \to taken(\tau))$$

Observation

An equivalent form is

$$\Box(\Box enabled(\tau) \to \Diamond taken(\tau)),$$

which looks more like the first-order formula we saw last time.

[1] weak and strong fairness will be "recurrent" (sorry for the pun) themes. For instance they will show up again in the TLA presentation.

Persistence

Definition 1.2.16 (Persistence)**.**

- A *persistence* formula is of the form

$$\Diamond\Box\varphi$$

  for some first-order formula φ.
- It states that all but finitely many positions satisfy φ[2]
- Persistence formulae are used to describe the eventual stabilization of some state property.

Recurrence and Persistence

Recurrence and persistence are duals.

$$\neg(\Box\Diamond\varphi) \sim (\Diamond\Box\neg\varphi)$$
$$\neg(\Diamond\Box\varphi) \sim (\Box\Diamond\neg\varphi)$$

Reactivity

Reactivity

Definition 1.2.17 (Reactivity)**.**

- A *simple reactivity* formula is of the form

$$\Box\Diamond\varphi \vee \Diamond\Box\psi$$

  for first-order formula φ and ψ.
- A very general class of formulae are conjunctions of reactivity formulae.
- An equivalent form is

$$\Box\Diamond\chi \to \Box\Diamond\psi,$$

  which states that if the computation contains infinitely many χ-positions, it must also contain infinitely many ψ-positions.

Reactivity

Proposition 4. *Strong fairness can be specified as the following reactivity formula.*

$$\Box\Diamond \mathit{enabled}(\tau) \to \Box\Diamond \mathit{taken}(\tau)$$

[2] In other words: only finitely ("but") many position satisfy $\neg\varphi$. So at some point onwards, it's always φ.

GCD Example

GCD Example

Below is a computation π of our recurring GCD program.

- a and b are fixed: $\pi \models \Box(a \doteq 21 \wedge b \doteq 49)$.
- $at(l)$ denotes the formulae $(\pi \doteq \{l\})$.
- *terminated* denotes the formula $at(l_8)$.

States are of the form $\langle \pi, x, y, g \rangle$.

$$\begin{aligned}
\pi : \quad & \langle l_1, 21, 49, 0\rangle \to \langle l_2^b, 21, 49, 0\rangle \to \langle l_6, 21, 49, 0\rangle \to \\
& \langle l_1, 21, 28, 0\rangle \to \langle l_2^b, 21, 28, 0\rangle \to \langle l_6, 21, 28, 0\rangle \to \\
& \langle l_1, 21, 7, 0\rangle \to \langle l_2^a, 21, 7, 0\rangle \to \langle l_4, 21, 7, 0\rangle \to \\
& \langle l_1, 14, 7, 0\rangle \to \langle l_2^a, 14, 7, 0\rangle \to \langle l_4, 14, 7, 0\rangle \to \\
& \langle l_1, 7, 7, 0\rangle \to \langle l_7, 7, 7, 0\rangle \to \langle l_8, 7, 7, 7\rangle \to \cdots
\end{aligned}$$

GCD Example

Does the following properties hold for π? And why?

<+(1)-> $\Box$*terminated* (safety)
<+(2)-> $at(l_1) \to$ *terminated*
<+(3)-> $at(l_8) \to$ *terminated*
<+(4)-> $at(l_7) \to \Diamond$*terminated* (conditional liveness)
<+(5)-> $\Diamond at(l_7) \to \Diamond$*terminated* (obligation)
<+(6)-> $\Box(\gcd(x, y) \doteq \gcd(a, b))$ (safety)
<+(7)-> $\Diamond$*terminated* (liveness)
<+(8)-> $\Diamond\Box(y \doteq \gcd(a, b))$ (persistence)
<+(9)-> $\Box\Diamond$*terminated* (recurrence)

1.2.9 Exercises

Exercises

1. Show that the following formulae are (not) LTL-valid.
   a) $\Box\varphi \leftrightarrow \Box\Box\varphi$
   b) $\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$
   c) $\neg\Box\varphi \to \Box\neg\Box\varphi$
   d) $\Box(\Box\varphi \to \psi) \to \Box(\Box\psi \to \varphi)$
   e) $\Box(\Box\varphi \to \psi) \vee \Box(\Box\psi \to \varphi)$
   f) $\Box\Diamond\Box\varphi \to \Diamond\Box\varphi$
   g) $\Box\Diamond\varphi \leftrightarrow \Box\Diamond\Box\Diamond\varphi$

2. A *modality* is a sequence of $\neg$, $\Box$ and $\Diamond$, including the empty sequence ϵ. Two modalities π and τ are *equivalent* if $\pi\varphi \leftrightarrow \tau\varphi$ is valid.
   a) Which are the non-equivalent modalities in LTL, and
   b) what are their relationship (ie. implication-wise)?

Chapter 2
References

Bibliography

[1] Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

[2] Manna, Z. and Pnueli, A. (1992). *The temporal logic of reactive and concurrent systems—Specification*. Springer Verlag, New York.

Index