

Expressiveness of Temporal Logics

(Outline of the lecture)

François Laroussinie and Nicolas Markey

Lab. Spécification & Vérification, CNRS & ENS Cachan, France

Abstract. This paper introduces some background on (propositional) temporal logics, as well as some of the expressiveness results that will be presented during the course. The material used during the lecture is available at <http://www.lsv.ens-cachan.fr/~markey/ESSLLI06/>.

1 Temporal Logics

Temporal logics are logics in which the truth values of formulas depend on the time at which they are evaluated. The ideas of temporal logics go back to Diodorus Cronus and Aristotle (around 400 B.C.), but temporal logics were formally defined and first studied by Arthur N. Prior, Saul Kripke and Hans W. Kamp in the 1950's and 1960's, as a special case of modal logics (where modalities are used to express that something will eventually hold, or will always hold in the future...) [1, 2].

The original works about temporal logics were carried out essentially with philosophy and linguistics in mind. In the late 1970's, Amir Pnueli [3] introduced temporal logics in computer science as a powerful tool to specify properties and reason about behaviours of reactive systems. Since then, temporal logics have received much attention, especially concerning its algorithmic issues. Many books and articles propose extensive surveys on these questions [4–8].

Different flavours of temporal logics coexist: the main two classes are *linear-time temporal logics*, where formulas are interpreted on one (linear) execution of the system, and *branching-time temporal logics*, where formulas concern all the possible evolutions of the system.

1.1 The linear-time framework

In the former class, LTL [9, 3] (for *(propositional) Linear-time Temporal Logic*) is the most famous logic. Its formulas are built from atomic propositions, boolean operators, and, usually, a two-place modality (read “until”). Formally,

$$\text{LTL} \ni \phi, \psi ::= \top \mid p \mid \phi \vee \psi \mid \phi \wedge \psi \mid \neg\phi \mid \phi \mathbf{U} \psi.$$

where p ranges over a (finite) set AP of atomic propositions. Formulas in LTL are interpreted along linear structures: a *linear structure* is a tuple $m = \langle L, <, l \rangle$, where $<$ is a total order over the infinite set L (assumed to have a minimal

element), and $l: L \rightarrow 2^{\text{AP}}$ assigns to each moment in time (i.e., to each element in L) a set of atomic propositions that hold true at that moment. The most relevant linear structures in this setting are \mathbb{R}^+ and \mathbb{N} , equipped with their usual order, since linear structures based on those sets can represent the behaviours of reactive systems (observed either continuously, or only at discrete moments). The semantics of LTL is defined inductively as follows: Given a model $\langle m, t \rangle$, made of a linear structure $m = \langle L, <, l \rangle$ and an element $t \in L$, for any atomic proposition p and LTLsubformulas ϕ and ψ , we have:

$$\begin{aligned}
\langle m, t \rangle \models \top & \text{ is always the case} \\
\langle m, t \rangle \models p & \text{ iff } p \in l(t) \\
\langle m, t \rangle \models \phi \vee \psi & \text{ iff } \langle m, t \rangle \models \phi \text{ or } \langle m, t \rangle \models \psi \\
\langle m, t \rangle \models \phi \wedge \psi & \text{ iff } \langle m, t \rangle \models \phi \text{ and } \langle m, t \rangle \models \psi \\
\langle m, t \rangle \models \neg\phi & \text{ iff it is not the case that } \langle m, t \rangle \models \phi \\
\langle m, t \rangle \models \phi \mathbf{U} \psi & \text{ iff } \exists u \in L. [t < u \text{ and } \langle m, u \rangle \models \psi \text{ and} \\
& \forall v \in L. ((t < v \text{ and } v < u) \Rightarrow \langle m, v \rangle \models \phi)]
\end{aligned}$$

Several linear-time temporal logics have been derived from LTL: for instance, LTL+Past is the extension of LTL with the modality **S** (read “since”), that is the symmetric counterpart of **U**. This extensions makes LTL closer to natural language, thus easier to use for practical applications. Restrictions of LTL have also been considered, since they might enjoy more efficient algorithms. For instance, the extra modality **F** (read “eventually”) is often defined with $\mathbf{F}\phi = \top \mathbf{U} \phi$, and the resulting logic (where the left-hand side formula of untils are required to be \top) is denoted with LTL_{**F**}. The dual modality **G**, defined by $\mathbf{G}\phi = \neg \mathbf{F} \neg\phi$, is read “always”.

1.2 The branching-time framework

On the other hand, CTL [10, 11] (for *Computation Tree Logic*) is the best-known logic in the branching-time framework. Branching-time logics are evaluated over tree orders: A *tree order* $<$ over an infinite set L is a partial order s.t., for any $t \in L$, the set $\{u \in L \mid u < t\}$ is totally ordered and has a least element. A branch is then a maximal, totally ordered subset of B . Again, the most relevant settings for computer scientists are those in which branches are isomorphic to \mathbb{R}^+ or \mathbb{N} , since they can be used to represent the computation trees of reactive systems. The course will essentially focus on discrete-time. A *branching structure* is then a tuple $m = \langle B, <, l \rangle$ where $\langle B, < \rangle$ is a tree order and $l: B \rightarrow 2^{\text{AP}}$ is a functions labeling each item of B with a set of atomic propositions that hold true at that point.

The syntax of CTL is given by the following grammar:

$$\text{CTL} \ni \phi, \psi ::= \top \mid p \mid \phi \vee \psi \mid \phi \wedge \psi \mid \neg\phi \mid \mathbf{E}\phi \mathbf{U} \psi \mid \mathbf{A}\phi \mathbf{U} \psi.$$

The semantics of **EU** and **AU** modalities, interpreted on a model $\langle m, t \rangle$ where t is a point in a branching structure $m = \langle B, <, l \rangle$, is then given by the following

second-order formulas (we omit the cases of atomic propositions and boolean operators, which are similar to the linear-time case):

$$\begin{aligned} \langle m, t \rangle \models \mathbf{E}\phi \mathbf{U} \psi \text{ iff } \exists b \subseteq B. [b \text{ is a branch of } B \text{ containing } t \wedge \\ \exists u \in b. [t < u \wedge (\langle m, u \rangle \models \psi) \wedge \\ \forall v \in b. ((t < v \wedge v < u) \Rightarrow \langle m, v \rangle \models \phi)] \\ \langle m, t \rangle \models \mathbf{A}\phi \mathbf{U} \psi \text{ iff } \forall b \subseteq B. [\text{if } b \text{ is a branch of } B \text{ containing } t, \text{ then} \\ \exists u \in b. [t < u \wedge (\langle m, u \rangle \models \psi) \wedge \\ \forall v \in b. ((t < v \wedge v < u) \Rightarrow \langle m, v \rangle \models \phi)]] \end{aligned}$$

Modalities \mathbf{EU} and \mathbf{AU} are read “exists until” and “for all until”, respectively. Intuitively, they express that an “until” formula (in the linear-time sense) holds along one or all the branches issued from the point where they are evaluated. That a subset of B is a branch can easily be expressed in first-order logic.

Again, many extensions of CTL have been defined and studied in the literature. In particular, the syntax of CTL forbids to separate the *path quantifier* (\mathbf{E} and \mathbf{A}) from the until modality. This requirement can be relaxed, yielding richer branching-time logics: CTL^+ is the extension where boolean combinations of (flat) until-formulas are allowed in the scope of path quantifiers, while CTL^* is the full extension, where \mathbf{U} -modalities can be nested. For instance, $\mathbf{E}(p \mathbf{U} q \wedge r \mathbf{U} s)$ is in both CTL^+ and CTL^* , but not (syntactically) in CTL, while $\mathbf{E}(p \mathbf{U}(q \mathbf{U} r))$ is in CTL^* but neither in CTL nor in CTL^+ .

2 Expressiveness of temporal logics

The notion of *expressiveness* of a temporal logic is an extension of the notion of *equivalence* of formulas. That two temporal logic formulas ϕ and ψ are *equivalent* over a set \mathcal{M} of models means that, for any model $m \in \mathcal{M}$, $m \models \phi$ if, and only if, $m \models \psi$. A temporal logic \mathcal{L} is said to be *more expressive* than a temporal logic \mathcal{L}' over a set of models \mathcal{M} if, for any formula $\phi \in \mathcal{L}'$, there is a formula $\psi \in \mathcal{L}$ that is equivalent to ϕ over \mathcal{M} . Two temporal logics are *equally expressive* when they are more expressive than each other.

Of course, the notion of equivalence, thus of expressiveness, can be extended to other formalisms, such as first- and second-order logics, automata on infinite words, etc. In this course, we present many important results about the relative expressiveness of different temporal logics, as well as results comparing the expressiveness of temporal logics and other formalisms. We only give a summary of those results here.

2.1 Expressiveness in the linear-time framework

One of the oldest result concerning the expressiveness of temporal logics is due to Hans W. Kamp:

Theorem 1 ([9]). *LTL+Past and first-order logic are equally expressive (over discrete models as well as over continuous ones).*

This result of course does not extend to LTL (without past). However, Dov M. Gabbay proved that, when restricting to the set of discrete-time initial models (i.e., models of the form $\langle m, 0 \rangle$ where 0 is the least element of m), the result can be extended:

Theorem 2 ([12, 13]). *LTL and first-order logic are equally expressive over discrete-time initial models. Moreover, over discrete-time models, formulas of LTL+Past can be separated, i.e., written as an equivalent boolean combination of pure-future (i.e., without the “since” modality) and pure-past (without the “until” modality) formulas.*

This immediately entails that LTL and LTL+Past are equally expressive over discrete-time initial models. The best known translation of LTL+Past formulas into LTL involves a quadruply-exponential blowup in the size of the formula, while only a singly-exponential succinctness gap has been proven to be unavoidable [14].

Other interesting expressiveness results that will be dealt with during the course include the tight link between linear-time temporal logics on the one hand, and Büchi automata and Büchi alternating automata on the other hand [15–19]. The course will also consider temporal logics with bounded nesting depth, and prove that they form a strict hierarchy [20, 21].

2.2 Expressiveness in the branching-time framework

The main expressiveness result concerning branching-time temporal logics concerns the relative expressiveness of several extensions of CTL:

Theorem 3. *Over discrete-time models, CTL^+ is as expressive [22] as CTL, and exponentially more succinct [23, 24]. Over discrete-time (and continuous-time) models, CTL^* is strictly more expressive than CTL [25].*

Many other extensions of CTL have been proposed, e.g. in order to cope with the fact that CTL cannot express *fairness* (i.e., the fact that something occurs infinitely often along some run) [25], or to add past-time modalities [26–28].

As in the part about linear-time, we will present the link between branching-time temporal logics and Büchi (tree) automata [29]. Several other topics will also be treated, including a characterization of formulas that can be expressed in both LTL and CTL, or the links between branching-time logics and behavioral equivalences (e.g. bi-simulation). Some results about ATL (*Alternating-time Temporal Logic*) [30, 31] will also be presented [32, 33].

2.3 Expressiveness of timed temporal logics

Timed temporal logics are extensions of (both linear-time and branching-time) temporal logics that include quantitative requirements on the elapse of time. In

the literature, two ways of achieving these extensions have been proposed: either by decorating modalities with an interval, or by using “clocks”. In the former case, the until modality now writes $\mathbf{U}_{[a,b]}$, with integers a and b , and means that the eventuality (i.e., the right-hand side subformula) must be witnessed within a and b time-units. The resulting logics are denoted by MTL (for *Metric Temporal Logic*) for the extension of LTL [34, 35], and TCTL (for *Timed CTL*) for the extension of CTL [37]. The other extension uses formula clocks, that are assumed to grow at the same rate as time, that can be reset before evaluating a subformula, and that can be compared to integer values. For instance, one can write

$$\mathbf{G}(p \Rightarrow x. \mathbf{F}(q \wedge x \leq 2))$$

which is read as follows: “it is always true that, when p holds, then we reset clock x , and eventually, $q \wedge x \leq 2$ will hold”. In other words, each time p holds, then q eventually holds within two time units. This would be expressed in MTL e.g. as $\mathbf{G}(p \Rightarrow \mathbf{F}_{[0,2]} q)$. This extension is called TPTL in the linear-time framework [39], and TCTL_c in the branching-time one [40].

The example above shows how TPTL can sometimes be translated into MTL. It turns out that such a translation is not always possible, and that TPTL is strictly more expressive than MTL, in both the discrete-time and continuous-time settings [41]. We will present those results, as well as some connections between timed temporal logics and timed (alternating) automata [42–45].

References

1. Antony Galton. *Temporal Logic*. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*, December 2003. <http://plato.stanford.edu/archives/win2003/entries/logic-temporal/>.
2. Wikipedia contributors. *Temporal Logic*. In Wikipedia, the Free Encyclopedia. May 2006. http://en.wikipedia.org/wiki/Temporal_logic
3. Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FoCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.
4. Dov M. Gabbay, Ian Hodkinson, and Mark A. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*. Clarendon Press, 1994.
5. Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, Philippe Schnoebelen. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer, January 2001.
6. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, January 2000.
7. E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 16, pages 995–1072. Elsevier, 1990.
8. Colin Stirling. Modal and temporal logics. In Samson Abramsky, Dov M. Gabbay, and Thomas S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 477–571. Oxford University Press, 1992.
9. Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, Los Angeles, California, USA, 1968.

10. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Proceedings of the 3rd Workshop on Logics of Programs (LOP'81)*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
11. Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Marangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings of the 5th International Symposium on Programming (SOP'82)*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 4 1982.
12. Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *Conference Record of the 7th ACM Symposium on Principles of Programming Languages (POPL'80)*, pages 163–173. ACM Press, 1980.
13. Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Proceedings of the Colloquium on Temporal Logic in Specification*, volume 398 of *Lecture Notes in Computer Science*. Springer, 1989.
14. François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Temporal logic with forgettable past. In *Proceedings of the 17th Annual Symposium on Logic in Computer Science (LICS'02)*, pages 383–392. IEEE Computer Society Press, 2002.
15. Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS'83)*, pages 185–194. IEEE Computer Society Press, 1983.
16. Moshe Y. Vardi. Nontraditional applications of automata theory. In Masami Hagiya and John C. Mitchell, editors, *Proceedings of the 2nd International Symposium on Theoretical Aspects of Computer Software (TACS'94)*, volume 789 of *Lecture Notes in Computer Science*, pages 575–597. Springer, 1994.
17. Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information & Computation*, 115(1):1–37, 1994.
18. Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.
19. Scott Rohde. *Alternating automata and the temporal logic of ordinals*. PhD thesis, University of Illinois, Urbana-Champaign, Illinois, USA, 1997.
20. Kousha Etessami and Thomas Wilke. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *Information & Computation*, 160(1-2):88–108, 2000.
21. Antonín Kučera and Jan Strejček. The stuttering principle revisited. *Acta Informatica*, 41(7-8):415–434, 2005.
22. E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Science*, 30(1):1–24, 1985.
23. Thomas Wilke. CTL⁺ is exponentially more succinct than CTL. In C. Pandu Rangan, Venkatesh Raman, and R. Ramanujam, editors, *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 1999.
24. Micah Adler and Neil Immerman. An $n!$ lower bound on formula size. In *Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01)*, pages 197–206. IEEE Computer Society Press, 2001.

25. E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.
26. Orna Kupferman and Amir Pnueli. Once and for all. In *Proceedings of the 10th Annual Symposium on Logic in Computer Science (LICS'95)*, pages 25–35. IEEE Computer Society Press, 1995.
27. François Laroussinie and Philippe Schnoebelen. A Hierarchy of Temporal Logics with Past. *Theoretical Computer Science* 148(2):303–324, 1995.
28. François Laroussinie and Philippe Schnoebelen. Specification in CTL+Past for verification in CTL. *Information and Computation* 156(1-2):236–263, 2000.
29. Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.
30. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 100–109. IEEE Computer Society Press, October 1997.
31. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
32. Valentin Goranko and Govert van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1-3):93–117, March 2006.
33. François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Expressiveness and complexity of ATL. Research Report LSV-06-03, Laboratoire Spécification et Vérification, ENS Cachan, France, February 2006. 20 pages.
34. Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
35. Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proc. 5th Annual Symposium on Logic in Computer Science (LICS'90)*, pages 390–401. IEEE Computer Society Press, 1990. Preliminary version of [36].
36. Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
37. Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Proc. 5th Annual Symposium on Logic in Computer Science (LICS'90)*, pages 414–425. IEEE Computer Society Press, 1990. Preliminary version of [38].
38. Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
39. Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.
40. Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model-checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
41. Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In R. Ramanujam and Sandeep Sen, editors, *Proceedings of the 25th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2005.
42. Rajeev Alur, Toms Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
43. Jean-François Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, University of Namur, Namur, Belgium, 1999.

44. Joël Ouaknine and James B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Annual Symposium on Logic in Computer Science (LICS'05)*. IEEE Computer Society Press, 2005.
45. Joël Ouaknine and James B. Worrell. On metric temporal logic and faulty Turing machines. In Holger Hermanns and Jens Palsberg, editors, *Proceedings of the 12th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*. Springer, 2006.