ELSEVIER

Information Processing Letters 54 (1995) 343-345

Information Processing Letters

About the expressive power of *CTL* combinators

F. Laroussinie [a,b,1]

[a] *LIFIA-IMAG, 46 Av. Félix Viallet, F-38000 Grenoble cedex, France*
[b] *Department of Computer Science, Aalborg University, Aalborg, Denmark*

Communicated by L. Kott; received 25 April 1994; revised 5 December 1994

Abstract

We present a new and quite surprising result about the expressive power of the $\exists\_\mathsf{U}$ and $\forall\_\mathsf{U}$ combinators in *CTL*.

Keywords: Temporal logic; *CTL*; Program specification; Specification languages

1. The *CTL* logic

CTL, the Computation Tree Logic proposed in [2] has been widely considered in literature for the specification of reactive systems [6,5]. *CTL* is paradigmatic in the field of branching-time temporal logic because it admits efficient model checking algorithms (see [3]) while remaining very expressive.

CTL formulas are built using four combinators: $\exists\mathsf{X}$, $\forall\mathsf{X}$, $\exists\_\mathsf{U}$ and $\forall\_\mathsf{U}$, plus atomic propositions $a, b, \ldots$ and boolean combinators:

$$
\begin{aligned}
&(CTL \ni)\ f, g ::= \\
&\quad \exists\mathsf{X}f \mid \forall\mathsf{X}f \mid \exists[f \,\mathsf{U}\, g] \mid \forall[f \,\mathsf{U}\, g] \\
&\quad \mid \neg f \mid f \wedge g \mid a \mid b \mid \ldots
\end{aligned}
$$

where we use the standard abbreviations: $\bot$, $\top$, $f \vee g$, $\ldots$ for resp. $a \wedge \neg a, \ldots$.

CTL formulas are interpreted over Kripke structures, i.e. directed graphs where every vertice (the *states*) carries a boolean valuation for the atomic propositions. See e.g. [5] for formal definitions. Informally, $\exists\mathsf{X}f$ means "there exists a next state satisfying f" so that a state q satisfies $\exists\mathsf{X}f$ (written $q \models \exists\mathsf{X}f$) iff for some q', a successor state of q (written $q \to q'$), it holds that $q' \models f$. $\forall\mathsf{X}$ means "for all next states ...", so that $\forall\mathsf{X}f \equiv \neg\exists\mathsf{X}\neg f$ (we do not allow states with no successors). $\exists[f \,\mathsf{U}\, g]$ means "from the current state, there is a run π satisfying f until g holds", that is a run $q_0 \to q_1 \to \cdots$ such that $q_k \models g$ for some k and $q_i \models f$ for all $0 \leqslant i < k$. $\forall[f \,\mathsf{U}\, g]$ means "from the current state, all runs satisfy f until g".

Additional useful operators are $\exists\mathsf{F}$ and $\forall\mathsf{F}$, which are weak versions of $\exists\_\mathsf{U}$ (resp. $\forall\_\mathsf{U}$): $\exists\mathsf{F}f$ means "from the current state, there is a (complete) run along which f will hold at some point", while $\forall\mathsf{F}f$ means "along all runs f will eventually hold". Clearly $\exists\mathsf{F}f$ can be defined as $\exists\top\mathsf{U}f$ and $\forall\mathsf{F}f$ as $\forall\top\mathsf{U}f$.

2. A folk result on *CTL*

In this setting it is well known that the $\forall\_\mathsf{U}$ operator can be expressed in terms of $\exists\_\mathsf{U}$ and $\forall\mathsf{F}$:

$$\forall[f \,\mathsf{U}\, g] \equiv \forall\mathsf{F}g \wedge \neg\exists[(\neg g) \,\mathsf{U}\, (\neg f \wedge \neg g)]. \tag{1}$$

[1] Email: fl@lifia.imag.fr.

This fact may be used to simplify proofs by induction over the structure of *CTL* formulas (see e.g. [1,8]), or to ease model-checking (see e.g. [3,4]).

In such cases ∀F is much simpler than ∀_U. ∃_U also is often simpler to deal with than ∀_U. Indeed, looking at the quantifier alternation in the semantic clause for ∀_U, we see that it has the general form "$\forall\ \pi\ \exists\ k \forall\ i \ldots$" and then is (syntactically) in Π_3, whereas (the clause for) ∃_U is in Σ_2 (witness "$\exists\ \pi\ \exists\ k\ \forall i$") and ∀F is in Π_2 (witness "$\forall\ \pi\ \exists k$"). Perhaps this explains why nobody (to our knowledge) has ever tried to express ∃_U in terms of ∀_U and ∃F even though many people think this is possible through a definition as simple as (1). We thought so for some time, but we were wrong.

In this note, we show that it is impossible to express ∃_U with ∀_U and ∃F. This result is quite surprising, especially when one knows about (1) and considers the position of the combinators in the alternation hierarchy. We believe the impossibility proof is interesting. Indeed every time we found (1) in the literature, no mention was made of the dual question, not even through an open question. By contrast, the question pops up every time we teach *CTL* in the classroom.

3. A surprising remark

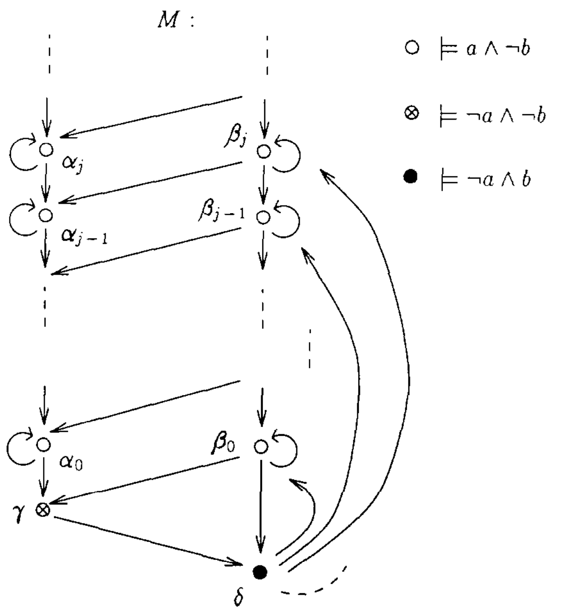
Eq. (1) entails the well-known result:

Folk Theorem 1. *CTL is no more expressive than the fragment L_E where only* ∃_U, ∀F, ∃X *are allowed.*

By contrast

Theorem 2. *CTL is strictly more expressive than the fragment L_A where only* ∃F, ∀_U, ∃X *are allowed. Specifically* $\exists[a \mathrm{U} b]$ *has no equivalent in L_A.*

For a proof, consider the Kripke structure M given in Fig. 1.



M :
α_j
α_{j-1}
α_0
β_j
β_{j-1}
β_0
γ
δ
○ $\models a \wedge \neg b$
⊗ $\models \neg a \wedge \neg b$
● $\models \neg a \wedge b$

Fig. 1. The valuation of the states are indicated by their color.

We prove that considering larger and larger j's, the states α_j and β_j agree on larger and larger L_A formulas. Formally, writing $|f|$ for the height of nested temporal operators in f, we have:

Lemma 3. *For all k and all $j \geqslant k$, α_j and β_j agree on any $f \in L_A$ such that $|f| \leqslant k$.*

Proof. By induction on k. The base case where $k = 0$ is clear: α_j and β_j agree on atomic propositions.

Now assume $k = k' + 1$: by the induction hypothesis α_j and β_j agree on all f such that $|f| < k$. Now consider some f with $|f| = k$. We prove that α_j and β_j agree on f by induction over the structure of f:

- The cases where f has the form $\neg g$ or $g_1 \wedge g_2$ are obvious.
- f is some $\exists \mathrm{X} g$ (and then $|g| \leqslant k'$): Assume $\beta_j \models \exists \mathrm{X} g$. There are three distinct cases depending on which successor of β_j satisfies g: it may be α_{j-1}, β_j or β_{j-1}. Then α_{j-1}, or by the induction hypothesis α_j or α_{j-1} satisfies g so that $\alpha_j \models \exists \mathrm{X} g$. The reverse direction (assuming $\alpha_j \models \exists \mathrm{X} g$) is dealt with in the same way.
- f is some $\exists \mathrm{F} g$: All reachable states from α_j are also reachable from β_j and conversely (because β_j is reachable from δ). Therefore α_j and β_j agree on f.
- f is some $\forall[g \mathrm{U} g']$: If $\alpha_j \models \forall[g \mathrm{U} g']$, then $\alpha_j \models g'$ because there is a run looping on α_j. By the induction hypothesis, $\beta_j \models g'$ and then $\beta_j \models \forall[g \mathrm{U} g']$. The reverse direction is proved in the same way. □

Now pick any L_A formula f and write k for $|f|$. The previous lemma states that if $j \geqslant k$, then $\alpha_j \models f$ iff $\beta_j \models f$. But $\alpha_j \not\models \exists[a \mathrm{U} b]$ and $\beta_j \models \exists[a \mathrm{U} b]$.

Therefore f is not equivalent to $\exists[a \mathsf{U} b]$ and this completes the proof of Theorem 2.

4. Variants and extensions

Clearly the above results apply to the logic *CTL*$\setminus$X, i.e. *CTL* without a "NextTime" operator, a logic suited to reasoning modulo stuttering [7].

Corollary 4.
- *CTL* $\setminus$ X *is no more expressive than the fragment where only* $\exists$_U *and* $\forall$F *are allowed.*
- *CTL*$\setminus$X *is strictly more expressive than the fragment where only* $\forall$_U *and* $\exists$F *are allowed.*

Another interesting logic considered in the literature is the *ECTL* logic from [6]. *ECTL* is "Extended *CTL*", or *CTL* with fairness. It contains the $\exists \overset{\infty}{\mathsf{F}}$ operator in addition to *CTL* operators. Informally $\exists \overset{\infty}{\mathsf{F}} f$ means "there is a run (starting from the current state) along which f holds infinitely often". (*ECTL* also has a $\forall \overset{\infty}{\mathsf{F}}$ combinator but $\forall \overset{\infty}{\mathsf{F}} f$ can be written as $\neg\exists\mathsf{F}\neg\forall\mathsf{F}f$.)

Corollary 5.
- *ECTL is no more expressive than the fragment where only* $\exists$_U, $\forall$F, $\exists \overset{\infty}{\mathsf{F}}$, $\exists$X *are allowed.*
- *ECTL is strictly more expressive than the fragment where only* $\forall$_U, $\exists$F, $\exists \overset{\infty}{\mathsf{F}}$ *and* $\exists$X *are allowed.*

Proof. It is easy to extend lemma 3 with $\exists \overset{\infty}{\mathsf{F}}$: for any run π starting from α_j (resp. β_j) there is a run π' starting from β_j (resp. α_j) and having π as a suffix. Clearly any property holding infinitely often along π also holds infinitely often along π'. $\square$

A well-known variant of the "Until" combinator is the "Unless" (see e.g. [5]), or "weak Until", written W, and differing from U in that a run satisfies $f\mathsf{W}g$ is f holds as long as g does not hold, but g is not required to hold eventually. Then *CTL* can be defined with W in place of U because one can express $\exists$_U and $\forall$_U with $\exists$_W and $\forall$_W (and conversely):

$$\exists[f \mathsf{U} g] \equiv \neg\forall[(\neg g)\mathsf{W}(\neg f \wedge \neg g)]$$
$$\exists[f\mathsf{W}g] \equiv \neg\forall[(\neg g) \mathsf{U} (\neg f \wedge \neg g)]$$
$$\forall[f \mathsf{U} g] \equiv \neg\exists[(\neg g)\mathsf{W}(\neg f \wedge \neg g)]$$
$$\forall[f\mathsf{W}g] \equiv \neg\exists[(\neg g) \mathsf{U} (\neg f \wedge \neg g)]$$

Then a consequence of our previous results is that *CTL* is strictly more expressive than the logic where only $\exists$_W, $\forall$_W$\perp$ and $\exists$X are allowed, while it is no more expressive than the logic where only $\forall$_W, $\exists$_W$\perp$ and $\exists$X are allowed.

Acknowledgements

We would like to thank Philippe Schnoebelen and Sophie Pinchinat for their many helpful suggestions and comments on this work.

References

[1] A. Bouajjani, R. Echahed and J. Sifakis, On model checking for real-time properties with durations, in: *Proc. 8th IEEE Symp. on Logic in Computer Science*, Montreal, 1993.
[2] E.M. Clarke and E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: *Proc. Logics of Programs Workshop*, Yorktown Heights, Lecture Notes in Computer Science **131** (Springer, Berlin, 1981) 52-71.
[3] E.M. Clarke, E.A. Emerson and A.P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, *ACM Trans. Programming Languages Systems* **8** (2) (1986) 244-263.
[4] E.M. Clarke and O. Grümberg, Research on automatic verification of finite-state concurrent systems, *Ann. Rev. Comput. Sci.* **2** (1987) 269-290.
[5] E.A. Emerson, Temporal and modal logic, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B*, Chapter 16 (Elsevier, Amsterdam, 1990) 995-1072.
[6] E.A. Emerson and J.Y. Halpern, "Sometimes" and "Not Never" revisited: On branching versus linear time temporal logic, *J. ACM* **33** (1) (1986) 151-178.
[7] L. Lamport, What good is temporal logic?, in: *Proc. Information Processing (IFIP) 83* (North-Holland, Amsterdam, 1983) 657-668.
[8] F. Laroussinie and Ph. Schnoebelen, A hierarchy of temporal logics with past, in: *Proc. STACS'94*, Caen, France, Lecture Notes in Computer Science **775** (Springer, Berlin, 1994) 47-58.