Extending fairness expressibility of ECTL$^+$: a tree-style one-pass tableau approach (Extended Abstract)

Alexander Bolotov *

University of Westminster, W1W 6UW, London, UK.

A.Bolotov@westminster.ac.uk

Montserrat Hermo Paqui Lucio†

University of the Basque Country, P. Manuel de Lardizabal, 1, 20018-San Sebastián, Spain.

montserrat.hermo@ehu.eus paqui.lucio@ehu.eus

Keywords: Temporal logic, fairness, tableau, branching-time, one-pass tableau.

In proceedings volume of 25th International Symposium on Temporal Representation and Reasoning (TIME 2018) as volume 120 of LIPIcs.

Temporal logic has become essential for the specification and verification of hardware and software systems. For the specification of the reactive and distributed systems, or, most recently, autonomous systems, the modelling of the possibilities 'branching' into the future is essential. Among important properties of these systems, so called fairness properties are important. In the standard formalisation of fairness, operators $\Diamond$ (eventually) and $\Box$ (always) have been used: $\mathsf{A}\Diamond\Box p$ – 'p' is true along all computation paths except possibly their finite initial interval, where 'A' is 'for all paths' quantifier, and $\mathsf{E}\Box\Diamond p$ – 'p' is true along a computation path at infinitely many states, where 'E' stands for 'there exists a path' quantifier. Branching-time logics (BTL) here give us an appropriate reasoning framework, where the most used class of formalisms are 'CTL' (Computation Tree Logic) type logics. CTL itself requires every temporal operator to be preceded by a path quantifier, thus, cannot express fairness. ECTL (Extended CTL) [5] enables simple fairness constraints but not their boolean combinations. ECTL$^+$ [6] further extends the expressiveness of ECTL allowing boolean combinations of temporal operators and ECTL fairness constraints (but not permitting their nesting). The logic CTL*, often considered as 'the full branching-time logic' overcomes these restrictions on expressing fairness. However, CTL* is extremely challenging for the application of any known technique of automated reasoning. Note that, unlike fair CTL [3] which, in tackling fairness, changes the underlying trees to those with 'fair paths' only, ECTL and ECTL$^+$ do not impose these changes.

From another perspective, the literature on fairness constraints, even in the linear-time setting, lacks the analysis of their formulation with the $\mathscr{U}$ ('until') operator. To the best of our knowledge, there are only a few research papers that raise or discuss the problem. For example, [10], introduces the logic LCTL, providing an extension of liveness constraints by the "until" operator. However, LCTL belongs to 'Fair CTL-type' logics [7]. 'Generalised liveness assumptions, which allow to express that

*The author would like to thank the University of Westminster for supporting the sabbatical in 2017

†These two authors have been partially supported by Spanish Projects TIN2013-46181-C2-2-R and TIN2017-86727-C2-2-R, and by the University of the Basque Country under Project LoRea GIU15/30.

Submitted to:
PROLE 2019

the conclusion $f_2 \mathscr{U} f_3$ of a liveness assumption $\Box(f_1 \Rightarrow (f_2 \mathscr{U} f_3))$ has to be satisfied' are addressed in [1]. The $\mathscr{U}$ operator in the formulation of the fairness can also be found in [14] which considers the sequential composition of processes, providing the following example - the composition of processes P_1 and P_2 'behaves as $P1$ until its termination and then behaves as $P2$'. Finally, [11] utilises restricted linear-time fairness constraints with $\mathscr{U}$ in the linear-time setting. We are not aware of any other analysis of fairness constraints in branching-time setting using the $\mathscr{U}$ operator and without restricting the underlying logic to be interpreted over the 'fair' paths. We bridge this gap, presenting the logic ECTL$^\#$ (we use # to indicate some restrictions on concatenations of the modalities and their boolean combinations). It is weaker than CTL* but extends ECTL$^+$ by allowing the combinations $\Box(A \mathscr{U} B)$ or $A \mathscr{U} \Box B$, referred to as modalities $\Box \mathscr{U}$ and $\mathscr{U} \Box$. This enables the formulation of stronger fairness constraints in the branching-time setting. The fairness constraint $\mathsf{A}(p \mathscr{U} \Box q)$ reads as 'q is true along all paths of the computation except possibly their finite initial interval, where p is true'. For example, in specifying the situation when a user of a system, changing passwords, cannot repeat any old password, the following property should hold: $\mathsf{A}((password = pw) \mathscr{U} \Box(password \neq pw))$ provided that pw is 'the current password'.

$\mathscr{B}(\mathscr{U}, \circ)$ (CTL) extensions	$\mathsf{E}(\Box\Diamond q)$	$\mathsf{E}(\Box\Diamond q \wedge \Box\Diamond r)$	$\mathsf{A}((p \mathscr{U} \Box q) \vee (s \mathscr{U} \Box \neg r))$	$\mathsf{A}\Diamond(\circ p \wedge \mathsf{E}\circ\neg p)$
$\mathscr{B}(\mathscr{U}, \circ, \Box\Diamond)$ (ECTL)	$\surd$	X	X	X
$\mathscr{B}^+(\mathscr{U}, \circ, \Box\Diamond)$ (ECTL$^+$)	$\surd$	$\surd$	X	X
$\mathscr{B}^+(\mathscr{U}, \circ, \mathscr{U}\Box)$ (ECTL$^\#$)	$\surd$	$\surd$	$\surd$	X
$\mathscr{B}^\star(\mathscr{U}, \circ)$ (CTL*)	$\surd$	$\surd$	$\surd$	$\surd$

Figure 1: Classification of CTL-type logics and their expressiveness

Figure 1 places our logic in the hierarchy of BTL representing their expressiveness: logics are classified by using '$\mathscr{B}$' for 'Branching', followed by the set of only allowed modalities as parameters; $\mathscr{B}^+$ indicates admissible boolean combinations of the modalities and $\mathscr{B}^\star$ reflects 'no restrictions' in either concatenations of the modalities or boolean combinations between them.[1] Thus, $\mathscr{B}(\mathscr{U}, \circ)$ denotes the logic CTL. In this hierarchy ECTL$^\#$ is $\mathscr{B}^+(\mathscr{U}, \circ, \mathscr{U}\Box)$.

For linear-time logics which deal with fairness in the linear-time setting, one-pass and two-pass tableau methods have been developed. In the repository of the CTL-type branching-time setting, the well-known logics ECTL and ECTL$^+$ were developed to explicitly deal with fairness. However, due to the syntactical restrictions, these logics can only express restricted versions of fairness. The logic CTL*, often considered as 'the full branching-time logic' overcomes these restrictions on expressing fairness. However, CTL* is extremely challenging for the application of verification techniques, and the tableau technique, in particular. For example, there is no one-pass tableau construction for CTL*, while one-pass tableau has an additional benefit enabling the formulation of dual sequent calculi that are often treated as more 'natural' being more friendly for human understanding. These two considerations lead to the following problem - are there logics that have richer expressiveness than ECTL$^+$, allowing the formulation of a new range of fairness constraints with 'until' operator, yet 'simpler' than CTL*, and for which a one-pass tableau can be developed? Here we give a positive answer to this question, introducing a sub-logic of CTL* called ECTL$^\#$, its tree-style one-pass tableau, and an algorithm for obtaining a systematic tableau, for any given admissible branching-time formulae. We prove the termination, soundness and

[1] This notation goes back to [4], here we use its nice tuning by Nicolas Markey in [12]. In the last column we use a short CTL* formula $\mathsf{A}\Diamond(\circ p \wedge \mathsf{E}\circ\neg p)$, not expressible by weaker logics. We found this formula indicative for CTL* as its validity is directly linked to the limit closure property [4].

completeness of the method. As tree-shaped one-pass tableaux are well suited for the automation and are amenable for the implementation and for the formulation of sequent calculi. Our results also open a prospect of relevant developments of the automation and implementation of the tableau method for $\mathsf{ECTL}^{\#}$, and of a dual sequent calculi.

We present a tree-style one-pass tableau for this logic continuing the analogous developments in linear-time case [2, 8] and for CTL [2]. An indicative feature of this approach is a context-based tableau technique. To the best of our knowledge, the context-based tableau has not been extended to BTL more expressive than CTL, though for them different other kinds of tableaux exist. In particular, [13] presents a tableau based decision procedure for $\mathsf{CTL}^{\star}$, which would definitely cover $\mathsf{ECTL}^{\#}$ as a sublogic of $\mathsf{CTL}^{\star}$. However, this tableau method is (unavoidably) complicated. For example, it utilises 'global conditions on infinite branches' to be checked by the automata-theoretic approach. Though such complications may be well justified by the complexity of $\mathsf{CTL}^{\star}$, aiming at a weaker logic, we would benefit by reducing the complications to the minimum. Also, a distinctive feature of the tableau method in [13] is the control of loops, specifically, of so called 'bad loops'. While it looks necessary for this technique, we would like to avoid similar complications for a simpler logic, $\mathsf{ECTL}^{\#}$. Moreover, due to the essential use of the notion of 'context' our tableau rules only produce 'good loops'. Tree-style one-pass tableaux (without additional procedures for checking meta-logical properties) have dual (cut-free) sequent calculi, see [8], enabling the construction of human-understandable proofs. In addition, these tableaux are well suited for the automation and are amenable for the implementation.[2] Our tableau is effectively an AND-OR tree where nodes are labelled by sets of state formulae. There are difficult cases of $\mathsf{ECTL}^{\#}$ formulae that appear due to the enriched syntax: disjunctions of formulae in the scope of the A quantifier and conjunctions of formulae in the scope of the E quantifier. To tackle these cases, in addition to $\alpha-\beta$ rules, that are standard to the tableaux, we use novel β^{+}-rules which use the context to force the eventualities to be fulfilled as soon as possible.

References

[1] Therese Berg & Harald Raffelt (2005): *Model Checking*. In Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker & Alexander Pretschner, editors: *Model-Based Testing of Reactive Systems*, Springer-Verlag, Berlin Heidelberg, pp. 557–603, doi:10.1007/b137241. Available at `https://www.springer.com/gb/book/9783540262787`.

[2] Kai Brünnler & Martin Lange (2008): *Cut-Free Sequent Systems for Temporal Logic*. *Journal of Logic and Algebraic Programming* 76(2), pp. 216–225, doi:10.1016/j.jlap.2008.02.004.

[3] Edmund M. Clarke, E. Allen Emerson & Aravinda P. Sistla (1986): *Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications*. *ACM Trans. Program. Lang. Syst.* 8(2), pp. 244–263, doi:10.1145/5397.5399.

[4] E. Allen Emerson (1990): *Temporal and Modal Logic*. In Jan van Leeuwen, editor: *Handbook of Theoretical Computer Science (Vol. B)*, MIT Press, Cambridge, USA, pp. 995–1072. Available at `http://dl.acm.org/citation.cfm?id=114891.114907`.

[5] E. Allen Emerson & Joseph Y. Halpern (1985): *Decision procedures and expressiveness in the temporal logic of branching time*. *Journal of Computer and System Sciences* 30(1), pp. 1 – 24, doi:10.1016/0022-0000(85)90001-7.

[6] E. Allen Emerson & Joseph Y. Halpern (1986): *Sometimes and Not Never Revisited: On Branching Versus Linear Time Temporal Logic*. *J. ACM* 33(1), pp. 151–178, doi:10.1145/4904.4999.

[2] An excellent survey of the seminal tableau techniques for temporal logics can be found in [9].

[7] E. Allen Emerson & Chin-Laung Lei (1986): *Temporal reasoning under generalized fairness constraints*. In Monien B. & Vidal-Naquet G., editors: *STACS 1986, Lecture Notes in Computer Science, vol 210*, Springer-Verlag Berlin Heidelberg, Karlsruhe, Federal Republic of Germany, pp. 21–37, doi:10.1007/3-540-16078-7_62.

[8] Jose Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro & Fernando Orejas (2009): *Dual Systems of Tableaux and Sequents for* PLTL. *Journal of Logic and Algebraic Programming* 78(8), pp. 701–722, doi:10.1016/j.jlap.2009.05.001.

[9] Rajeev Gore (1999): *Tableau Methods for Modal and Temporal Logics*. In Marcello D'Agostino, Dov M. Dov Gabbay, Reiner Hähnle & Joachim Posegga, editors: *Handbook of Tableau Methods*, Springer, Netherlands, Dordrecht, pp. 297–396, doi:10.1007/978-94-017-1754-0_6.

[10] Bernhard Josko (1987): *Model checking of CTL formulae under liveness assumptions*. In Thomas Ottmann, editor: *Automata, Languages and Programming, 14th International Colloquium*, Springer-Verlag Berlin Heidelberg, Karlsruhe, Federal Republic of Germany, pp. 5–24, doi:10.1007/3-540-18088-5.

[11] Jan Kretinsky & Ruslan Ledesma Garza (2013): *Rabinizer 2: Small Deterministic Automata for LTL\GU*. In: *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013*, Springer, Heidelberg Dordrecht London New York, pp. 446–450, doi:10.1007/978-3-319-02444-8_32.

[12] Nicolas Markey (2013): *Temporal Logics*. Available at `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/NM-coursTL13.pdf`. Course notes, Master Parisien de Recherche en Informatique, Paris, France.

[13] Mark Reynolds (2009): *A Tableau for CTL**. In Ana Cavalcanti & Dennis Dams, editors: *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings*, *Lecture Notes in Computer Science* 5850, Springer, pp. 403–418, doi:10.1007/978-3-642-05089-3_26.

[14] Jun Sun, Yang Liu, Jin Song Dong & Hai H. Wang (2008): *Specifying and Verifying Event-based Fairness Enhanced Systems*. In Shaoying Liu, Tom Maibaum & Keijiro Araki, editors: *Formal Methods and Software Engineering: 10th International Conference on Formal Engineering Methods ICFEM 2008*, Springer Science and Business Media, Kitakyushu-City, Japan, pp. 5–24, doi:10.1007/978-3-540-88194-0.