Validity of Contextual Formulas

Javier Esparza
Technical University of Munich, Germany

Rubén Rubio
Universidad Complutense de Madrid, Spain

Abstract

Many well-known logical identities are naturally written as equivalences between *contextual formulas*. A simple example is the Boole-Shannon expansion $c[p] \equiv (p \wedge c[\text{true}]) \vee (\neg p \wedge c[\text{false}])$, where c denotes an arbitrary formula with possibly multiple occurrences of a "hole", called a *context*, and $c[\varphi]$ denotes the result of "filling" all holes of c with the formula φ. Another example is the unfolding rule $\mu X.c[X] \equiv c[\mu X.c[X]]$ of the modal μ-calculus.

We consider the modal μ-calculus as overarching temporal logic and, as usual, reduce the problem whether $\varphi_1 \equiv \varphi_2$ holds for contextual formulas φ_1, φ_2 to the problem whether $\varphi_1 \leftrightarrow \varphi_2$ is valid. We show that the problem whether a contextual formula of the μ-calculus is valid for all contexts can be reduced to validity of ordinary formulas. Our first result constructs a *canonical context* such that a formula is valid for all contexts iff it is valid for this particular one. However, the ordinary formula is exponential in the nesting-depth of the context variables. In a second result we solve this problem, thus proving that validity of contextual formulas is EXP-complete, as for ordinary equivalences. We also prove that both results hold for CTL and LTL as well. We conclude the paper with some experimental results. In particular, we use our implementation to automatically prove the correctness of a set of six contextual equivalences of LTL recently introduced by Esparza et al. for the normalization of LTL formulas. While Esparza et al. need several pages of manual proof, our tool only needs milliseconds to do the job and to compute counterexamples for incorrect variants of the equivalences.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases μ-calculus, temporal logic, contextual rules

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.24

Related Version *Extended Version*: `arXiv:2407.07759` [8]

Supplementary Material *Software (Source Code)*: `https://github.com/ningit/ctxform` [12]

Funding This work was partially supported by the Agencia Estatal de Investigación (AEI) under project PID2019-108528RB-C22.

1 Introduction

Some well-known identities useful for reasoning in different logics can only be easily formulated as *contextual identities*. One example is the Boole-Shannon expansion of propositional logic, which constitutes the foundation of Binary Decision Diagrams and many SAT-solving procedures [1]. It can be formulated as

$$c[p] \equiv (p \wedge c[\text{true}]) \vee (\neg p \wedge c[\text{false}]) \tag{1}$$

where, intuitively, c denotes a Boolean formula with "holes", called a *context*, and $c[\varphi]$ denotes the result of "filling" every hole of the context c with the formula φ. For example, if $c := ([\,] \wedge p) \vee (q \rightarrow [\,])$ and $\varphi := p$, then $c[p] = (p \wedge p) \vee (q \rightarrow p)$. More precisely, c is a *context variable* ranging over contexts, and the equivalence sign $\equiv$ denotes that for all possible assignments of contexts to c the ordinary formulas obtained on both sides of $\equiv$ are equivalent.

$$
\begin{array}{rcl}
c[\psi_1 \,\mathbf{U}\, \psi_2] \,\mathbf{W}\, \varphi & \equiv & (\mathbf{GF}\, \psi_2 \wedge c[\psi_1 \,\mathbf{W}\, \psi_2] \,\mathbf{W}\, \varphi) \vee c[\psi_1 \,\mathbf{U}\, \psi_2] \,\mathbf{U}\, (\varphi \vee \mathbf{G}\, c[\text{false}]) \\
\varphi \,\mathbf{W}\, c[\psi_1 \,\mathbf{U}\, \psi_2] & \equiv & \varphi \,\mathbf{U}\, c[\psi_1 \,\mathbf{U}\, \psi_2] \vee \mathbf{G}\, \varphi \\
c[\mathbf{GF}\, \psi] & \equiv & (\mathbf{GF}\, \psi \wedge c[\text{true}]) \vee c[\text{false}] \\
c[\mathbf{FG}\, \psi] & \equiv & (\mathbf{FG}\, \psi \wedge c[\text{true}]) \vee c[\text{false}] \\
\mathbf{GF}\, c[\psi_1 \,\mathbf{W}\, \psi_2] & \equiv & \mathbf{GF}\, c[\psi_1 \,\mathbf{U}\, \psi_2] \vee (\mathbf{FG}\, \psi_1 \wedge \mathbf{GF}\, c[\text{true}]) \\
\mathbf{FG}\, c[\psi_1 \,\mathbf{U}\, \psi_2] & \equiv & (\mathbf{GF}\, \psi_2 \wedge \mathbf{FG}\, c[\psi_1 \,\mathbf{W}\, \psi_2]) \vee \mathbf{FG}\, c[\text{false}]
\end{array}
$$

Figure 1 Rewrite system for the normalization of LTL formulas [9].

For linear-time temporal logic in negation normal form, a useful identity similar to the Boole-Shannon expansion is

$$c[\mathbf{GF}\, p] \equiv (\mathbf{GF}\, p \wedge c[\text{true}]) \vee c[\text{false}] \tag{2}$$

where c now ranges over formulas of LTL with holes. For example, the identity shows that $q \,\mathbf{U}\, (\mathbf{GF}\, p \wedge r)$ is equivalent to $\mathbf{GF}\, p \wedge q \,\mathbf{U}\, r \vee q \,\mathbf{U}\, \text{false}$, and so after simplifying equivalent to $\mathbf{GF}\, p \wedge q \,\mathbf{U}\, r$.[1] As a third example, the unfolding rule of the μ-calculus (the fundamental rule in Kozen's axiomatization of the logic [2]

$$\mu X.c[X] \equiv c[\mu X.c[X]]$$

whose formulation requires nested contexts. Further examples of contextual LTL identities are found in [9], where, together with Salomon Sickert, we propose a rewrite system to transform arbitrary LTL formulas into formulas of the syntactic fragment Δ_2, with at most single-exponential blowup.[2] The rewrite system consists of the six identities (oriented from left to right) shown in Figure 1.

Remarkably, to the best of our knowledge the *automatic* verification of contextual equivalences like the ones above has not been studied yet. In particular, we do not know of any automatic verification procedure for any of the identities above. In [9] we had to prove manually that the left and right sides of each identity are equivalent for every context (Lemmas 5.7, 5.9, and 5.11 of [9]), a tedious and laborious task; for example, the proof of the first identity alone takes about 3/4 of a page. This stands in sharp contrast to non-contextual equivalences, where an ordinary equivalence $\varphi_1 \equiv \varphi_2$ of LTL can be automatically verified by constructing a Büchi automaton for the formula $\neg(\varphi_1 \leftrightarrow \varphi_2)$ and checking its emptiness. So the question arises whether the equivalence problem for contextual formulas is decidable, and in particular whether the manual proofs of [9] can be replaced by an automated procedure. In this paper we give an affirmative answer.

Let σ be a mapping assigning contexts to all context variables of a formula φ, and let $\sigma(\varphi)$ denote the ordinary formula obtained by instantiating φ with σ. The equivalence, validity, and satisfiability problems for contextual formulas are:

---

1 The restriction to formulas in negation normal form is necessary. For example, taking $c := \neg(p \wedge [\,])$ does not yield a valid equivalence.

2 Δ_2 contains the formulas in negation normal form such that every path of the syntax tree exhibits at most one alternation of the strong and weak until operators $\mathbf{U}$ and $\mathbf{W}$. They have different uses, and in particular they are easier to translate into deterministic ω-automata [9].

1. *Equivalence* of φ_1 and φ_2: does $\sigma(\varphi_1) \equiv \sigma(\varphi_2)$ hold for every σ?
2. *Validity* of φ: is $\sigma(\varphi)$ valid (in the ordinary sense) for every σ?
3. *Satisfiability* of φ: is $\sigma(\varphi)$ satisfiable (in the ordinary sense) for some σ?

We choose the modal μ-calculus as overarching logic, and prove that these problems can be reduced to their counterparts for ordinary μ-calculus formulas. As corollaries, we also derive reductions for CTL and LTL. More precisely, we obtain the following two results.

First result. Given a contextual formula φ with possibly multiple occurrences of a context variable c, there exists a *canonical instantiation* κ_φ of c, also called the *canonical context*, such that φ is valid/satisfiable iff the ordinary formula $\kappa_\varphi(\varphi)$ is valid/satisfiable. Further, κ_φ can be easily computed from φ by means of a syntax-guided procedure.

To give a flavour of the idea behind the canonical instantiation consider the distributive law $\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$ for ordinary Boolean formulas. It is well-known that such a law is correct iff it is correct for the special case in which $\varphi_1, \varphi_2, \varphi_3$ are distinct Boolean variables, say p_1, p_2, p_3. In other words, the law is correct iff the Boolean formula $p_1 \wedge (p_2 \vee p_3) \leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$ is valid. This result does not extend to contextual formulas. For example, consider the contextual equivalence (2), reformulated as the validity of the contextual formula

$$\varphi := c[\mathbf{GF}\, p] \;\leftrightarrow\; \big((\mathbf{GF}\, p \wedge c[\text{true}]) \vee c[\text{false}]\big) \tag{3}$$

While φ is valid, the ordinary formula $\varphi^d := p_1 \leftrightarrow \big((\mathbf{GF}\, p \wedge p_2) \vee p_3\big)$ obtained by replacing $c[\mathbf{GF}\, q]$, $c[\text{true}]$, and $c[\text{false}]$ by atomic propositions p_1, p_2, p_3, respectively, is not. (We call φ^d the *decontextualization* of φ.) Loosely speaking, the replacement erases dependencies between $c[\mathbf{GF}\, q]$, $c[\text{true}]$, and $c[\text{false}]$. For example, since contexts are formulas in negation normal form, $c[\text{false}] \models c[\text{true}]$ or $c[\text{false}] \models c[\mathbf{GF}\, p]$ hold for every context c, but we do not have $p_3 \models p_2$ and $p_2 \models p_1$. To remedy this, we choose a context κ_φ that informally states:

> At every moment in time, p_1 holds if the hole is filled with a formula globally entailing $\mathbf{GF}\, p$ and p_2 holds if it is filled with a formula globally entailing true and p_3 holds if it is filled with a formula globally entailing false.

The context is:

$$\kappa_\varphi := \mathbf{G}\left(\big(\mathbf{G}\,([\,] \to \mathbf{GF}\, p) \to p_1\big) \wedge \big(\mathbf{G}\,([\,] \to \text{true}) \to p_2\big) \wedge \big(\mathbf{G}\,([\,] \to \text{false}) \to p_3\big)\right) \tag{4}$$

Our result shows that κ_φ is a canonical context for φ. In other words, the contextual formula φ of (3) is valid iff the ordinary formula

$$\kappa_\varphi(\varphi) := \;\kappa_\varphi[\mathbf{GF}\, p] \;\leftrightarrow\; \big((\mathbf{GF}\, p \wedge \kappa_\varphi[\text{true}]) \vee \kappa_\varphi[\text{false}]\big)$$

obtained by setting $c := \kappa_\varphi$ in (3), is valid. After substituting according to (4) and simplifying, we obtain

$$\kappa_\varphi(\varphi) \equiv \big(\mathbf{G}\,(p_1 \wedge p_2) \wedge \mathbf{G}\,\big(\mathbf{GF}\, p \vee p_3\big)\big) \leftrightarrow \big(\mathbf{GF}\, p \wedge \mathbf{G}\,((\mathbf{GF}\, p \to p_1) \wedge p_2) \vee \mathbf{G}\,(p_1 \wedge p_2 \wedge p_3)\big) \tag{5}$$

So (3) is valid iff the formula on the right-hand-side of (5) is valid, which is proved by SPOT 2.11 [4] in milliseconds.

Second result. Given a contextual formula φ, the ordinary formula $\kappa_\varphi(\varphi)$ has $O(|\varphi|^d)$ length, where d is the nesting depth of the context variables. Since $d \in O(n)$, the blowup is exponential. Our second result provides a polynomial reduction. Let $c[\psi_1]$, ..., $c[\psi_n]$ be the context expressions appearing in φ. Instead of finding a canonical instantiation, we focus on adding to the decontextualized formula φ^d information on the dependencies between $c[\psi_1]$, ..., $c[\psi_n]$. For every pair ψ_i, ψ_j, we add to φ^d the premise $\mathbf{G}\left(\mathbf{G}\left(\psi_i \to \psi_j\right) \to (p_i \to p_j)\right)$. Intuitively, the premise "transforms" dependencies between ψ_i and ψ_j into dependencies between fresh atomic propositions p_i and p_j. For example, we obtain that (3) is valid iff the ordinary LTL formula

$$\mathbf{G}\left(\bigwedge_{i=1}^{3}\bigwedge_{j=1}^{3}\left(\mathbf{G}\left(\psi_i \to \psi_j\right) \to (p_i \to p_j)\right)\right) \to \left(p_1 \leftrightarrow ((\mathbf{GF}\, p \wedge p_2) \vee p_3\right)$$

is valid or, after simplification, iff

$$\mathbf{G}\left(\begin{array}{c}(\mathbf{FG}\,\neg p \to (p_1 \to p_3)) \wedge (\mathbf{GF}\, p \to (p_2 \to p_1)) \\ \wedge \\ (p_3 \to p_1) \wedge (p_1 \to p_2)\end{array}\right) \to \left(p_1 \leftrightarrow ((\mathbf{GF}\, p \wedge p_2) \vee p_3)\right) \quad (6)$$

is valid. Again, Spot 2.11 proves that (6) is valid within milliseconds. Since the premise has polynomial size in the size of the original contextual formula, we obtain a reduction from contextual validity to ordinary validity with polynomial blowup. Observe, however, that the ordinary formula is not obtained by directly instantiating the context variable c.

Experiments. We have implemented our reductions and connected them to validity and satisfiability checkers for propositional logic (PySAT [10] and MiniSat [5]), LTL (Spot 2.11 [4]), and CTL (CTL-SAT [11]). We provide some experimental results. In particular, we can prove the correctness of all the LTL identities of [9] within milliseconds.

Structure of the paper. Section 2 recalls the standard μ-calculus and presents its contextual extension. Section 3 studies the validity problem of contextual propositional formulas, as an appetizer for the main results on the μ-calculus in Section 4. These are extended to CTL and LTL in Sections 4.4 and 4.5. Experimental results are presented in Section 5, and Section 6 gives some conclusions.

2 The contextual μ-calculus

We briefly recall the syntax and semantics of the μ-calculus [2], and then introduce the syntax and semantics of the contextual μ-calculus.

The modal μ-calculus. The syntax of the modal μ-calculus over a set AP of atomic propositions and a set V of variables is

$$\varphi ::= p \mid \neg p \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\cdot\rangle\, \varphi \mid [\cdot]\, \varphi \mid \mu X.\varphi \mid \nu X.\varphi \quad (7)$$

where $p \in AP$ and $X \in V$. The semantics is defined with respect to a Kripke structure and a valuation. A *Kripke structure* is a tuple $\mathcal{K} = (S, \to, I, AP, \ell)$, where S and I are sets of states and initial states, $\to \subseteq S \times S$ is the transition relation (where we assume that every state has at least one successor), and $\ell \colon S \to \mathcal{P}(S)$ assigns to each state a set of atomic propositions. A *valuation* is a mapping $\eta\colon V \to \mathcal{P}(S)$. Given $\mathcal{K}$ and η, the semantics assigns to each formula φ a set $[\![\varphi]\!]_\eta \subseteq S$, the set of states satisfying φA. A Kripke structure $\mathcal{K}$ satisfies φ if $I \subseteq [\![\varphi]\!]_\eta$. The mapping $[\![\cdots]\!]_\eta$ is inductively defined by:

$$
\begin{aligned}
[\![p]\!]_\eta &= \{s \in S \mid p \in \ell(s)\} & [\![\langle\cdot\rangle\,\varphi]\!]_\eta &= \{s \in S \mid \exists s'.s \to s' \wedge s' \in [\![\varphi]\!]_\eta\} \\
[\![\neg p]\!]_\eta &= S \setminus [\![p]\!]_\eta & [\![[\cdot]\,\varphi]\!]_\eta &= \{s \in S \mid \forall s'.s \to s' \Rightarrow s' \in [\![\varphi]\!]_\eta\} \\
[\![X]\!]_\eta &= \eta(X) & & \\
[\![\varphi_1 \wedge \varphi_2]\!]_\eta &= [\![\varphi_1]\!]_\eta \cap [\![\varphi_2]\!]_\eta & [\![\mu X.\varphi]\!]_\eta &= \bigcap\{U \subseteq S \mid [\![\varphi]\!]_{\eta[X/U]} \subseteq U\} \\
[\![\varphi_1 \vee \varphi_2]\!]_\eta &= [\![\varphi_1]\!]_\eta \cup [\![\varphi_2]\!]_\eta & [\![\nu X.\varphi]\!]_\eta &= \bigcup\{U \subseteq S \mid U \subseteq [\![\varphi]\!]_{\eta[X/U]}\}
\end{aligned}
$$

Let $\mathrm{FV}(\varphi) \subseteq V$ be the set of free variables, i.e. not bound by a fixpoint operator, in a formula φ. Observe that if φ is a closed formula (that is, $\mathrm{FV}(\varphi) = \emptyset$), then $[\![\varphi]\!]_\eta$ depends only on $\mathcal{K}$, not on η, so we just write $[\![\varphi]\!]$. On the contrary, when dealing with multiple Kripke structures at the same time, we write $[\![\varphi]\!]_{\mathcal{K},\eta}$ or $[\![\varphi]\!]_{\mathcal{K}}$ to avoid ambiguity. We say that $\mathcal{K}$ satisfies φ, denoted $\mathcal{K} \models \varphi$, if $I \subseteq [\![\varphi]\!]_\eta$, that is, if every initial state satisfies φ. A closed formula φ is valid (satisfiable) if $\mathcal{K} \models \varphi$ for every (some) Kripke structure $\mathcal{K}$.

It is well-known that every formula of the μ-calculus is equivalent to a formula in which all occurrences of a variable are either bound or free, and every two distinct fixpoint subformulas have different variables.

The contextual modal μ-calculus. *Contextual formulas* are expressions over a set AP of atomic propositions, a set V of variables, and a set C of *context variables*. (The contextual formulas of the introduction only had one contextual variable, but in general they can have multiple and arbitrarily nested variables.) They are obtained by extending the syntax (7) with a new term:

$$\varphi ::= p \mid \neg p \mid X \mid \cdots \mid \nu X.\varphi \mid c[\varphi]$$

where $c \in C$. For the semantics, we need to introduce contexts and their instantiations. A *context* is an expression over the syntax that extends (7) with *holes*:

$$\varphi ::= p \mid \neg p \mid X \mid \cdots \mid \nu X.\varphi \mid [\ ]$$

We let $\mathcal{C}$ denote the set of all contexts. An *instantiation* of the set C of context variables is a mapping $\sigma\colon C \to \mathcal{C}$. Given a contextual formula φ, we let $\sigma(\varphi)$ denote the ordinary formula obtained as follows: in the syntax tree of φ, proceeding bottom-up, repeatedly replace each expression $c[\psi]$ by the result of filling all holes of the context $\sigma(c)$ with ψ. Here is a formal inductive definition:

▶ **Definition 1** (instantiation). *Let $\mathbb{F}$ and $\mathbb{C}$ be the sets of ordinary and contextual formulas of the μ-calculus. An* instantiation *is a function $\sigma : C \to \mathcal{C}$ binding each context variable to a context. We lift an instantiation σ to a mapping $\bar{\sigma}_c : \mathbb{C} \to \mathbb{F}$ as follows:*

1. $\bar{\sigma}_c(p) = p$.
2. $\bar{\sigma}_c(c[\varphi]) = (\sigma(c))[[\ ]/\bar{\sigma}_c(\varphi)]$ *(i.e., the result of substituting $\bar{\sigma}_c(\varphi)$ for $[\ ]$ in $\sigma(c)$).*
3. $\bar{\sigma}_c(\varphi_1 \wedge \varphi_2) = \bar{\sigma}_c(\varphi_1) \wedge \bar{\sigma}_c(\varphi_2)$.
4. $\bar{\sigma}_c(\varphi_1 \vee \varphi_2) = \bar{\sigma}_c(\varphi_1) \vee \bar{\sigma}_c(\varphi_2)$.
5. $\bar{\sigma}_c(\langle\cdot\rangle\,\varphi) = \langle\cdot\rangle\,\bar{\sigma}_c(\varphi)$.
6. $\bar{\sigma}_c([\cdot]\,\varphi) = [\cdot]\,\bar{\sigma}_c(\varphi)$.
7. $\bar{\sigma}_c(\mu X.\varphi) = \mu X.\bar{\sigma}_c(\varphi)$.
8. $\bar{\sigma}_c(\nu X.\varphi) = \nu X.\bar{\sigma}_c(\varphi)$.

Abusing language, we overload σ and write $\sigma(\varphi)$ for $\bar{\sigma}(\varphi)$.

▶ **Example 2.** Let $\varphi = p\,\mathbf{U}\,c_1[\,(p \vee c_1[q])\,\mathbf{W}\,c_2[\neg p \vee q)]\,]$. Further, let $\sigma(c_1) := \mathbf{G}\,[\,]$ and $\sigma(c_2) := ([\,] \wedge q)$. We have $\sigma(\varphi) = p\,\mathbf{U}\,\mathbf{G}\,\big((p \vee \mathbf{G}\,q)\,\mathbf{W}\,((\neg p \vee q) \wedge q)\big)$.

We can now extend the notions of validity and satisfaction from ordinary to contextual formulas.

▶ **Definition 3** (validity and satisfiability)**.** *A closed contextual formula φ is valid if $\mathcal{K} \models \sigma(\varphi)$ for every instantiation σ and Kripke structure $\mathcal{K}$, and satisfiable if $\mathcal{K} \models \sigma(\varphi)$ for some instantiation σ and Kripke structure $\mathcal{K}$.*

3 Validity of contextual propositional formulas

As an appetizer, we study the validity and satisfiability problems for the propositional fragment of the modal μ-calculus, which allows us to introduce the main ideas in the simplest possible framework. The syntax of contextual propositional formulas over sets AP and C of propositional and contextual variables is

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid c[\varphi] \tag{8}$$

where $p \in AP$ and $c \in C$. The semantics is induced by the semantics of the modal μ-calculus, but we quickly recall it. Given a *valuation* $\beta\colon AP \to \{0,1\}$, the semantics of an ordinary formula φ is the Boolean $[\![\varphi]\!]_\beta \in \{0,1\}$, defined as usual, e.g. $[\![\varphi_1 \wedge \varphi_2]\!]_\beta = 1$ iff $[\![\varphi_1]\!]_\beta = 1$ and $[\![\varphi_2]\!]_\beta = 1$. Given a valuation β and an instantiation $\sigma\colon C \to \mathcal{C}$ of the context variables, the semantics of a contextual formula is the boolean $[\![\sigma(\varphi)]\!]_\beta$, where $\sigma(\varphi)$ is the formula obtained by instantiating each context variable c with the context $\sigma(c)$.

We will use the substitution lemma of propositional logic. Let $\mathbb{F}$ and $\mathbb{C}$ be the set of all ordinary and contextual propositional formulas, respectively.

▶ **Lemma 4** (substitution lemma)**.** *For any $\varphi \in \mathbb{F}$, valuation $\beta\colon AP \to \{0,1\}$, and substitution $\sigma\colon AP \to \mathbb{F}$, $[\![\sigma(\varphi)]\!]_\beta = [\![\varphi]\!]_{\beta'}$ where β' is given by $\beta'(p) := [\![\sigma(p)]\!]_\beta$ for every $p \in V$.*

Moreover, since formulas with syntax (8) are in negation normal form, we have the following monotonicity result.

▶ **Lemma 5** (monotonicity)**.** *For any $\varphi, \psi, \psi' \in \mathbb{F}$, propositional variable p that does not appear negated in φ, and valuation $\beta\colon AP \to \{0,1\}$, if $[\![\psi \to \psi']\!]_\beta = 1$ then $[\![\varphi[p/\psi] \to \varphi[p/\psi']]\!]_\beta = 1$.*

3.1 Canonical instantiations

Let us now prove that a contextual propositional formula $\varphi \in \mathbb{C}$ is valid (satisfiable) iff it is valid (satisfiable) for the following *canonical instantiation* κ_φ of its context variables.

▶ **Definition 6** (maximal context subformulas)**.** *A* context subformula *of $\varphi \in \mathbb{C}$ is a subformula of φ of the form $c[\psi]$ for some $c \in C$ and $\psi \in \mathbb{C}$. The set of context subformulas of φ is denoted $CSub(\varphi)$. A context subformula is* maximal *if it is not a proper subformula of any other context subformula. The* decontextualization *of φ, denoted φ^d, is the result of replacing every maximal context subformula $c[\psi]$ of φ by a fresh propositional variable $p_{c[\psi]}$.*

▶ **Definition 7** (canonical instantiation of a contextual formula)**.** *The* canonical instantiation *of $\varphi \in \mathbb{C}$, also called the* canonical context*, is the mapping $\kappa_\varphi\colon C \to \mathcal{C}$ that assigns to every context variable $c \in C$ the context*

$$\kappa_\varphi(c) := \bigwedge_{c[\psi] \in CSub(\varphi)} \big([\,] \to \psi^d\big) \to p_{c[\psi]}$$

▶ **Example 8.** Let us illustrate Definition 7 on an example. Boole-Shannon's expansion holds iff the contextual formula

$$\varphi := c[p] \leftrightarrow \big((p \wedge c[\mathrm{true}]) \vee (\neg\, p \wedge c[\mathrm{false}])\big)$$

is valid. We have $\mathit{CSub}(\varphi) = \{c[p], c[\mathrm{true}], c[\mathrm{false}]\}$. All elements of $\mathit{CSub}(\varphi)$ are maximal. Since $p^d = p$, $\mathrm{true}^d = \mathrm{true}$, and $\mathrm{false}^d = \mathrm{false}$, the canonical instantiation is given by

$$\kappa_\varphi(c) = \big(([\,] \to p) \to p_{c[p]}\big) \wedge \big(([\,] \to \mathrm{true}) \to p_{c[\mathrm{true}]}\big) \wedge \big(([\,] \to \mathrm{false}) \to p_{c[\mathrm{false}]}\big)$$

We need an auxiliary lemma.

▶ **Lemma 9.** *For any $\varphi \in \mathbb{C}$, instantiation σ, and valuation β, there is a valuation β' that coincides with β in every variable occurring in $\sigma(\varphi)$ and satisfies $[\![\sigma(\varphi)]\!]_\beta = [\![\kappa_\varphi(\varphi)]\!]_{\beta'}$.*

Proof sketch (full proof in [8]). Given φ, σ, and β, we define $\beta'(p_{c[\psi]}) = [\![\sigma(c[\psi])]\!]_\beta$ for every $c[\psi] \in \mathit{CSub}(\varphi)$ and $\beta'(p) = \beta(p)$ otherwise. We first prove $[\![\sigma(\phi)]\!]_\beta = [\![\phi^d]\!]_{\beta'}$ as a direct application of the substitution lemma with $\gamma_\sigma(p_{c[\psi]}) = \sigma(c[\psi])$, which satisfies $\gamma_\sigma(\phi^d) = \sigma(\phi)$. Then, we prove $[\![\sigma(\phi)]\!]_\beta = [\![\kappa_\varphi(\phi)]\!]_{\beta'}$ by induction on ϕ, using the previous statement and some calculations on the expression of $\kappa_\varphi(c[\psi])$ using the monotonicity of contexts by Lemma 5. ◀

▶ **Proposition 10** (fundamental property of the canonical instantiation)**.** *A contextual formula $\varphi \in \mathbb{C}$ is valid (resp. satisfiable) iff the ordinary formula $\kappa_\varphi(\varphi) \in \mathbb{F}$ is valid (resp. satisfiable).*

Proof. For validity, if φ is valid, then $\sigma(\varphi)$ is valid for every instantiation σ, and so in particular $\kappa_\varphi(\varphi)$ is valid. For the other direction, assume $\kappa_\varphi(\varphi)$ is valid. We prove that $\sigma(\varphi)$ is also valid for any instantiation σ. Let β be a valuation. By Lemma 9 there is another valuation β' such that $[\![\sigma(\varphi)]\!]_\beta = [\![\kappa(\varphi)]\!]_{\beta'}$. Moreover, we have $[\![\kappa(\varphi)]\!]_{\beta'} = 1$ because $\kappa(\varphi)$ is valid. So $\sigma(\varphi)$ is valed, because β is arbitrary.

Satisfiability is handled by a dual proof. If $\kappa_\varphi(\varphi)$ is satisfiable, then so is φ by definition. If φ is satisfiable, then there is an instantiation σ such that $[\![\sigma(\varphi)]\!]_\beta = 1$. Lemma 9 give us a valuation β' such that $[\![\kappa_\varphi(\varphi)]\!]_{\beta'} = [\![\sigma(\varphi)]\!]_\beta = 1$. ◀

▶ **Example 11.** Let φ and $\kappa_\varphi(c)$ be as in Example 8. By definition, we have $\kappa_\varphi(\varphi) := \kappa_\varphi(c[p]) \leftrightarrow \big((p \wedge \kappa_\varphi(c[\mathrm{true}])) \vee (\neg\, p \wedge \kappa_\varphi(c[\mathrm{false}]))\big)$. Simplification yields

$$\kappa_\varphi(\varphi) \quad \equiv \quad \begin{array}{c} p_{c[p]} \wedge p_{c[\mathrm{true}]} \wedge (\neg p \to p_{c[\mathrm{false}]}) \\ \leftrightarrow \\ (p \wedge (p \to p_{c[p]}) \wedge p_{c[\mathrm{true}]}) \vee (\neg p \wedge p_{c[p]} \wedge p_{c[\mathrm{true}]} \wedge p_{c[\mathrm{false}]}) \end{array}$$

This formula is not valid, and so by Proposition 10 Boole-Shannon's expansion is valid.

The following example shows that the ordinary formula $\kappa_\varphi(\varphi)$ may be exponentially larger than the contextual formula φ when φ contains nested contexts.

▶ **Example 12.** Consider the contextual formula $\varphi := c^n[q]$, where $c^0[\psi] := \psi$ and $c^n[\psi] := c[c^{n-1}[\psi]]$ for every formula ψ. The size of φ is $n + 4$. The canonical context is $\kappa_\varphi(c) = \bigwedge_{l=1}^n ([\,] \to p_{c^{l-1}[q]}) \to p_{c^l[q]}$ with $p_{c^0[q]} = q$. Instantiating the "holes" of $\kappa_\varphi(c)$ with a formula ψ of size k yields the formula $\kappa_\varphi(c)[[\,]/\psi]$ of size $n(7 + k) - 1 \geq nk$. Since $\kappa_\varphi(\varphi) = \kappa_\varphi(c^n[q]) = \kappa(c)[[\,]/\kappa(c^{n-1}[q])]$ by definition, the size of $\kappa_\varphi(\varphi)$ is at least $n! = (|\varphi| - 4)!$, and so exponential in the size of φ.

3.2 A polynomial reduction

As anticipated in the introduction, in order to avoid the exponential blowup illustrated by the previous example, we consider a second method that relies on finding an ordinary formula equivalid to the contextual formula. This will lead us to the complexity result of Corollary 14.

▶ **Proposition 13.** *A propositional contextual formula $\varphi \in \mathbb{C}$ is valid iff the ordinary propositional formula*

$$\varphi_e := \left(\bigwedge_{c[\psi_1], c[\psi_2] \in CSub(\varphi)} (\psi_1^d \to \psi_2^d) \to (p_{c[\psi_1]} \to p_{c[\psi_2]}) \right) \to \varphi^d$$

is valid.

Proof sketch (full proof in [8]). We follow here the same ideas of Section 3.1. ($\Rightarrow$) If φ_e is valid, for a given substitution σ and Kripke structure $\mathcal{K}$, we define the Kripke structure $\mathcal{K}'$ of Lemma 9. After showing again that $[\![\phi^d]\!]_{\beta'} = [\![\sigma(\phi)]\!]_\beta$ for every subformula, we see that the condition of φ_e holds through a calculation, and then its conclusion yields $[\![\varphi^d]\!]_{\beta'} = [\![\sigma(\varphi)]\!]_\beta = 1$, so φ is valid. ($\Leftarrow$) If φ is valid, so is $\kappa_\varphi(\varphi)$ with a valuation β. We show φ_e holds under the same valuation. This is immediate if the premise does not hold. Otherwise, we can use the monotonicity encoded in the premise of φ_e to almost repeat the calculation on $\kappa_\varphi(c[\psi])$ in Lemma 9 and conclude $[\![\kappa_\varphi(\varphi)]\!]_\beta = [\![\varphi^d]\!]_{\beta'} = 1$. ◀

▶ **Corollary 14.** *The validity and satisfiability problems for contextual propositional formulas are co-NP-complete and NP-complete, respectively.*

Proof. Proposition 13 gives a polynomial reduction to validity of ordinary formulas. Indeed, $|\varphi^d| \leq |\varphi| + |\varphi|^3 \cdot (2|\varphi| + 5) \leq 8|\varphi|^4$. For satisfiability, it suffices to replace the top implication of φ_e by a conjunction. ◀

▶ Remark 15. In the propositional calculus, once we assign truth values to the atomic propositions every formula is equivalent to either true or false. Similarly, every context is equivalent to true, false, or [ ]. Hence, an alternative method to check validity of a contextual propositional formula is to check the validity of all possible instantiations of the context variables with these three contexts. However, for n different context variables, this requires 3^n validity checks.

▶ Remark 16. Other examples of valid identities are $c[p \wedge q] \equiv c[p] \wedge c[q]$, $c[p \vee q] \equiv c[p] \vee c[q]$, and $c[p] \equiv c[c[p]]$. Example of valid entailments are $(p \leftrightarrow q) \models (c[p] \leftrightarrow c[q])$ and $(p \to q) \models (c[p] \to c[q])$; the entailments in the other direction are not valid, as witnessed by the instantiation $\sigma(c) :=$ false. Finally, $p \equiv c[p]$ is an example of an identity that is not valid in any direction. All these facts can be automatically checked using any of the methods described in the section.

4 Validity of contextual μ-calculus formulas

We extend the reductions of Section 3 to the contextual modal μ-calculus. In particular, this requires introducing a new definition of canonical instantiation and a new equivalid formula. The main difference with the propositional case is that contexts may now contain free variables (that is, variables that are bound outside the context). For example, in the unfolding rule we find the context $c[X]$, and X appears free in the argument of c. This problem will be solved by replacing each free variable X by either the fixpoint subformula that binds it, or by a fresh atomic proposition p_X. We will also need to tweak decontextualizations. More precisely, the canonical instantiation will have the shape

$$\kappa_\varphi(c) := \bigwedge_{c[\psi] \in CSub(\varphi)} (\mathbf{AG}\,([\ ] \to \psi^*)) \to p_{c[\psi]}$$

where $\mathbf{AG}\,\psi$ is an abbreviation for $\nu X.([\cdot]X \wedge \psi)$, and ψ^* is a slight generalization of ψ^d.

Throughout the section we let $\mathbb{F}$ and $\mathbb{C}$ denote the sets of all ordinary and contextual formulas of the contextual μ-calculus over sets AP, V, and C, of atomic propositions, variables, and context variables, respectively. Further, we assume w.l.o.g. that all occurrences of a variable in a formula are either bound or free, and that distinct fixpoint subformulas have distinct variables. The following notation is also used throughout:

▶ **Definition 17.** *Given a formula $\varphi \in \mathbb{C}$ and a bound variable X occurring in φ, we let $\alpha X.\varphi_X$ denote the unique fixpoint subformula of φ binding X.*

4.1 Variable and propositional substitutions

A key tool to obtain the results of Section 3 was the substitution lemma for propositional logic. On top of atomic propositions, the μ-calculus has also variables, and we need separate substitution lemmas for both of them. We start with the variable substitution lemma. In this case, we have a μ-calculus formula φ with some free variables $X \in \mathrm{FV}(\varphi)$ and we want to replace them by closed formulas $\sigma(X) \in \mathbb{F}$. As usual, bound variables are not replaced by variable substitutions, i.e. $\sigma(\alpha X.\phi) = \alpha.\sigma|_{V\setminus\{X\}}(\phi)$. The following lemma is a direct translation of the substitution lemma for propositional logic.

▶ **Lemma 18** (variable substitution lemma). *For any Kripke structure with set of states S, valuation $\eta : V \to \mathcal{P}(S)$, and substitution $\sigma : V \to \mathbb{F}$ such that $\sigma(X)$ is either X or a closed formula for all $X \in V$, we have $[\![\sigma(\varphi)]\!]_\eta = [\![\varphi]\!]_{\eta'}$ where η' is defined by $\eta'(X) := [\![\sigma(X)]\!]_\eta$.*

Replacing atomic propositions is more subtle, since they can be mapped to non-ground μ-calculus formulas. While propositions have a fixed value, the semantics of their replacements may depend on the valuation, which could be the dynamic result of fixpoint calculations appearing in the formula. Consequently, the substitution lemma will not work for an arbitrary valuation like in Lemma 18, but only for those who *match* the values of the fixpoint variables of the formula.

▶ **Definition 19** (fixpoint valuation). *η is a* fixpoint valuation *of a formula $\varphi \in \mathbb{F}$ iff $\eta(X) = [\![\alpha X.\phi]\!]_\eta$ for every bound variable X of φ.*

▶ **Lemma 20** (propositional substitution lemma). *For any Kripke structure $\mathcal{K} = (S, \to, I, AP, \ell)$, formula $\varphi \in \mathbb{F}$, substitution $\sigma : AP \to \mathbb{F}$ such that $\sigma(p) = p$ if p appears negated in φ, and valuation $\eta : V \to \mathcal{P}(S)$ such that $\eta(X) = [\![\alpha X.\sigma(\phi_X)]\!]_\eta$ for every subformula $\alpha X.\phi_X$ of φ, we have $[\![\sigma(\varphi)]\!]_{\mathcal{K},\eta} = [\![\varphi]\!]_{\mathcal{K}',\eta}$ where $\mathcal{K}' = (S, \to, AP, I, \ell')$ is the Kripke structure with $\ell'(p) = [\![\sigma(p)]\!]_\eta$ for every $p \in AP$.*

These two lemmas will be very helpful in the proof of the main theorems, where we turn subformulas and variables into atomic propositions to make formulas like $\kappa_\varphi(\varphi)$ and φ_e ordinary and closed, respectively. For example, consider $\mu X.c[X] = c[\mu X.c[X]]$. If we take ψ^d instead of ψ^* in the canonical instantiation, we obtain

$$\kappa_\varphi(c) = \big((\mathbf{AG}\,([\ ] \to X)) \to p_{c[X]}\big) \wedge \big((\mathbf{AG}\,([\ ] \to \mu X.p_{c[X]})) \to p_{c[\mu X.c[X]]}\big) \tag{9}$$

Since X is free in this context, which value should it take? The following lemma proves there is a unique fixpoint valuation $\hat{\eta}$ for each formula φ and Kripke structure $\mathcal{K}$. This will give the answer to this question.

▶ **Lemma 21** (existence of fixpoint valuation). *For every Kripke structure $\mathcal{K}$ and closed formula $\varphi \in \mathbb{F}$, there is a unique fixpoint valuation $\hat{\eta}$, up to variables that do not appear in φ.*

Moreover, in (9), we will be interested in getting rid of the free occurrence of X to reduce the problem to validity of ordinary closed formulas. The following lemma claims that we can replace the free variables in the formula by the fixpoint subformula defining those variables, without changing the semantics of the formula. We call the substitution achieving this, which we show to be independent of any Kripke structure, the *fixpoint substitution* of φ.

▶ **Lemma 22** (fixpoint substitution). *For every closed formula $\varphi \in \mathbb{F}$, there is a (unique) variable substitution $\hat{\sigma} : X \to \mathbb{F}$ such that $\hat{\sigma}(X)$ is closed and $\hat{\sigma}(X) = \hat{\sigma}(\alpha X.\phi_X)$. Moreover, for every Kripke structure $\mathcal{K}$, every subformula ϕ of φ, and every fixpoint valuation η for φ, we have $[\![\sigma(\phi)]\!] = [\![\phi]\!]_\eta$.*

4.2 Canonical instantiation

We are now ready to define the canonical instantiation of a contextual formula.

▶ **Definition 23** (canonical instantiation of a contextual formula). *Let $\varphi \in \mathbb{C}$ be a contextual formula of the μ-calculus. Given a subformula ψ of φ, let ψ^* be the result of applying to ψ^d its fixpoint substitution. The* canonical instantiation *of φ is the mapping $\kappa_\varphi : C \to \mathcal{C}$ defined by*

$$\kappa_\varphi(c) := \bigwedge_{c[\psi] \in CSub(\varphi)} (\mathbf{AG}\,([\ ] \to \psi^*)) \to p_{c[\psi]}$$

where $\mathbf{AG}\,\psi$ is an abbreviation for $\nu X.([\cdot]X \wedge \psi)$ for some fresh variable X.

We prove that φ is valid (resp. satisfiable) iff $\kappa_\varphi(\varphi)$ is valid (satisfiable). We need two lemmas. The first one is the extension of Lemma 5 to the μ-calculus.

▶ **Lemma 24** (monotonicity). *For every $\varphi, \psi, \psi' \in \mathbb{F}$, where only φ may contain $[\ ]$, fixpoint valuation $\hat{\eta}$, and every $s \in S$, if $s \in [\![\mathbf{AG}\,(\psi \to \psi')]\!]_\eta$, then $s \in [\![\varphi[[\ ]/\psi]]\!]_{\hat{\eta}}$ implies $s \in [\![\varphi[[\ ]/\psi']]\!]_{\hat{\eta}}$.*

The second lemma is the key one.

▶ **Lemma 25.** *For every $\varphi \in \mathbb{F}$, instantiation σ, and Kripke structure $\mathcal{K} = (S, \to, I, AP, \ell)$, there is a Kripke structure $\mathcal{K}' = (S, \to, AP', I, \ell')$, where $AP \subseteq AP'$ and ℓ' extends ℓ, such that $[\![\sigma(\varphi)]\!]_{\mathcal{K}} = [\![\kappa_\varphi(\varphi)]\!]_{\mathcal{K}'}$.*

Proof sketch (full proof in [8]). The ideas of Lemma 9 are reproduced here, although with the additional complication of μ-calculus variables. Given φ, σ, and $\mathcal{K}$, we define $\mathcal{K}' := (S, \to, I, AP', \ell')$, $AP' := AP \cup \{p_{c[\psi]} \mid c[\psi] \in CSub(\varphi)\}$, $\ell'(p) = \ell(p)$ if $p \in AP$, and $\ell'(p_{c[\psi]}) = [\![\sigma(c[\psi])]\!]_{\hat{\eta}}$ using the fixpoint valuation $\hat{\eta}$ of Lemma 21 for $\sigma(\varphi)$. First, we show $[\![\phi^*]\!]_{\hat{\eta}} = [\![\phi^d]\!]_{\hat{\eta}}$ for every subformula ϕ of φ using the variable substitution lemma (Lemma 18) with $\gamma_*(X) = \alpha X.\varphi_X$. Then, like in the proposition case, we prove $[\![\sigma(\phi)]\!]_{\hat{\eta}} = [\![\phi^d]\!]_{\hat{\eta}}$ for any subformula ϕ using the propositional substitution lemma (Lemma 20) with $\gamma_\sigma(p_{c[\psi]}) = \sigma(c[\psi])$. Finally, we prove $[\![\sigma(\phi)]\!]_{\hat{\eta}} = [\![\kappa_\varphi(\phi)]\!]_{\hat{\eta}}$ by induction using some calculation (essentially the same in Lemma 9) by the monotonicity of Lemma 24 on the expression of $\kappa_\varphi(c)$ as well as the propositional substitution lemma. ◀

▶ **Theorem 26.** *For every $\varphi \in \mathbb{C}, \varphi$ is valid (resp. satisfiable) iff $\kappa_\varphi(\varphi) \in \mathbb{F}$ is valid (satisfiable).*

Proof. For validity: ($\Rightarrow$) If φ is valid, then $\sigma(\varphi)$ is valid for every instantiation σ. In particular, $\kappa_\varphi(\varphi)$ is valid. ($\Leftarrow$) Assuming $\kappa_\varphi(\varphi)$ is valid and for any instantiation σ, we must prove that $\sigma(\varphi)$ is also valid. Let $\mathcal{K}$ be a Kripke structure, Lemma 25 claims there is another Kripke structure $\mathcal{K}'$ such that $[\![\sigma(\varphi)]\!]_{\mathcal{K}} = [\![\kappa_\varphi(\varphi)]\!]_{\mathcal{K}'}$, so $\mathcal{K} \models \sigma(\varphi)$ iff $\mathcal{K}' \models \kappa_\varphi(\varphi)$. Moreover, we have $\mathcal{K}' \models \kappa_\varphi(\varphi)$ because $\kappa_\varphi(\varphi)$ is valid, so $\sigma(\varphi)$ is valid too since $\mathcal{K}$ is arbitrary.

For satisfiability, ($\Leftarrow$) If $\kappa_\varphi(\varphi)$ is satisfiable, then φ is satisfiable by definition. ($\Rightarrow$) If φ is satisfiable, then $\mathcal{K} \models \sigma(\varphi)$ for some σ and $\mathcal{K}$. Lemma 25 then ensures $\mathcal{K}' \models \kappa_\varphi(\varphi)$ for some $\mathcal{K}'$, so $\kappa_\varphi(\varphi)$ is satisfiable. ◀

As in the propositional case, the length of $\kappa_\varphi(\varphi)$ grows exponentially in the nesting depth of the context. However, in the μ-calculus it can also grow exponentially even for non-nested contexts. The reason is that applying the fixpoint substitution to a μ-formula can yield an exponentially larger formula.

▶ **Example 27.** Consider $\varphi = \mu X_1. \cdots \mu X_n.X_1 \wedge \cdots \wedge X_n$ for any $n \in \mathbb{N}$, whose size is $|\varphi| = 3n - 1$. It can be proven by induction that $|\sigma(X_k)| = 2^{k-1}(3n-2)+1$, so $|\sigma(X_n)| = 2^{n-1}(3n-2)+1 = 2^{\frac{1}{3}(|\varphi|-2)}(|\varphi|+1)+1 \in O(2^{|\varphi|})$.[3]

Both problems are solved in the next section by giving an alternative reduction to validity/satisfiability of ordinary μ-calculus formulas.

4.3 A polynomial reduction

Given a contextual formula φ of the μ-calculus, we construct an ordinary formula φ_e equivalid to φ of polynomial size in φ. Following the idea of Proposition 13, we replace all context occurrences in φ by fresh atomic propositions, and insert additional conditions to ensure that the values of these propositions are consistent with what they represent. However, in the μ-calculus, these conditions may introduce unbound variables, and the strategy to remove them in Theorem 26 involves the exponential blowup attested by Example 27. To solve this problem, we also replace the free μ-calculus variables in the context occurrences by fresh atomic propositions; further, we add additional clauses to ensure that they take a value consistent with the fixpoint calculation.

▶ **Definition 28** (equivalid formula). *For every contextual formula $\varphi \in \mathbb{C}$,*
- *let $F = \bigcup_{c[\psi] \in CSub(\varphi)} \mathrm{FV}(\psi)$ be the free variables in all context arguments of φ;*
- *for every $X \in F$, let p_X be a fresh variable that does not occur in φ; and*
- *for every subformula ϕ of φ, let ϕ^+ be the result of replacing every free occurrence of X in the formula ϕ^d by p_X.*

We define the ordinary formula $\varphi_e \in \mathbb{F}$ as

$$\left(\bigwedge_{\substack{c[\psi_1], c[\psi_2] \\ \in CSub(\varphi)}} \mathbf{AG}\,(\mathbf{AG}\,(\psi_1^+ \to \psi_2^+) \to (p_{c[\psi_1]} \to p_{c[\psi_2]})) \wedge \bigwedge_{X \in F} \mathbf{AG}\,(p_X \leftrightarrow \alpha X.\phi_X^+) \right) \to \varphi^+$$

We say $\mathcal{K}'$ is an extension of $\mathcal{K}$ if $\mathcal{K}' = (S, \{\to\}, I, AP', \ell')$, $AP \subseteq AP'$, and $\ell'|_{AP} = \ell$.

[3] The difficulty in the previous example are fixpoint formulas with free variables. Otherwise, $|\sigma(X)| = |\sigma(\alpha X.\varphi_X)| = |\alpha X.\varphi_X| \leq |\varphi|$.

▶ **Proposition 29.** *For every contextual formula $\varphi \in \mathbb{C}$, instantiation σ, and Kripke structure $\mathcal{K} = (S, \rightarrow, I, AP, \ell)$,*

1. *there is an extension $\mathcal{K}'$ of $\mathcal{K}$ such that $\mathcal{K} \models \sigma(\varphi)$ iff $\mathcal{K}' \models \varphi_e$.*
2. *for any extension $\mathcal{K}'$ of $\mathcal{K}$ such that $\mathcal{K}'$ is a model for the premise of φ_e, $\mathcal{K}' \models \kappa_\varphi(\varphi)$ iff $\mathcal{K}' \models \varphi_e$.*

Proof sketch (full proof in [8]). We follow the main strategy of Proposition 13. (1) Using the valuation $\hat{\eta}$ given by Lemma 21 for $\sigma(\varphi)$, we define the Kripke structure $\mathcal{K}'$ of Lemma 25 with some more variables $\ell'(p_X) = [\![\alpha X.\varphi_X]\!]_{\mathcal{K},\hat{\eta}}$. Again, for every subformula, we prove $[\![\phi^d]\!]_{\mathcal{K}',\hat{\eta}} = [\![\phi^+]\!]_{\mathcal{K}',\hat{\eta}}$, $[\![\phi^d]\!]_{\mathcal{K}',\hat{\eta}} = [\![\sigma(\phi)]\!]_{\mathcal{K},\hat{\eta}}$ invoking the appropriate substitution lemmas. This let us prove that the premise of φ_e holds because of the monotonicity of contexts reflected in Lemma 24 and the matching definitions of $\hat{\eta}$ and $\ell'(p_X)$. Hence, φ_e is equivalent to its conclusion, and we have proven $[\![\varphi^d]\!]_{\mathcal{K}',\hat{\eta}} = [\![\sigma(\varphi)]\!]_{\mathcal{K},\hat{\eta}}$, which implies the statement.

(2) Let $\varphi_e = \varphi_p \rightarrow \varphi^d$, we now assume $\mathcal{K}' \models \varphi_p$ and must prove $\mathcal{K}' \models \kappa_\varphi(\varphi)$ iff $\mathcal{K}' \models \varphi_e$, or equivalently iff $\mathcal{K}' \models \varphi^+$. Using the fixpoint valuation $\hat{\eta}$ of $\kappa_\varphi(\varphi)$, we inductively extract from the premise that $\ell'(p_X) = \hat{\eta}(X)$ and $\ell'(p_{c[\psi]}) = [\![\kappa_\varphi(c[\psi])]\!]_{\hat{\eta}}$ with the usual calculations and substitutions. Then, we conclude that the right argument of the top implication in φ_e satisfies $[\![\varphi^+]\!] = [\![\sigma(\varphi)]\!]$, which implies the statement. ◀

▶ **Theorem 30.** *A contextual μ-calculus formula $\varphi \in \mathbb{C}$ is valid iff $\varphi_e \in \mathbb{F}$ is valid.*

Proof. ($\Leftarrow$) φ is valid if $\sigma(\varphi)$ is valid for every instantiation σ. Let $\mathcal{K}$ be any Kripke structure, $\mathcal{K} \models \sigma(\varphi)$ must hold. However, Proposition 29 ensures there exists $\mathcal{K}'$ such that $\mathcal{K} \models \sigma(\varphi)$ if $\mathcal{K} \models \varphi_e$. Since φ_e is valid, we are done. ($\Rightarrow$) If φ is valid, so is $\kappa_\varphi(\varphi)$. We should prove that φ_e is valid, which means $\mathcal{K} \models \varphi_e$ for all $\mathcal{K}$. If the premise of φ_e does not hold, $\mathcal{K} \models \varphi_e$ trivially. Otherwise, Proposition 29 reduce the problem to $\mathcal{K} \models \kappa_\varphi(\varphi)$, which holds by hypothesis. ◀

▶ **Corollary 31.** *The validity and satisfiability problems for contextual μ-calculus formulas are* EXPTIME*-complete.*

Proof. The validity and satisfiability problems for ordinary formulas of the μ-calculus are EXPTIME-complete [2]. Theorem 30 gives a polynomial reduction from contextual to ordinary validity. Indeed, a rough bound is $|\varphi_e| \leq |\varphi|^3 \cdot (2|\varphi| + 5) + |\varphi| \cdot (|\varphi| + 2) + |\varphi| \leq 11|\varphi|^4$. For satisfiability, we can again replace the top implication of φ_e by a conjunction. ◀

4.4 Validity of contextual CTL formulas

We assume that the reader is familiar with the syntax and semantics of CTL (see e.g. [3]), and only fix a few notations. The syntax of contextual CTL over a set AP of atomic propositions is:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid c[\varphi] \mid \mathbf{A}\,(\varphi\,\mathbf{U}\,\varphi) \mid \mathbf{A}\,(\varphi\,\mathbf{W}\,\varphi) \mid \mathbf{E}\,(\varphi\,\mathbf{U}\,\varphi) \mid \mathbf{E}\,(\varphi\,\mathbf{W}\,\varphi)$$

where $p \in AP$, $c \in C$, and $\mathbf{U}$, $\mathbf{W}$ are the strong until and weak until operators. As for the μ-calculus, the syntax of contextual CTL-formulas adds a term $c[\varphi]$, and the syntax of CTL contexts adds the hole term [ ].

Given a Kripke structure $\mathcal{K} = (S, \rightarrow, I, AP, \ell)$ and a mapping $\ell\colon S \rightarrow \mathcal{P}(S)$, the semantics assigns to each ordinary formula φ the set $[\![\varphi]\!] \subseteq S$ of states satisfying φ. For example, $[\![\mathbf{E}\,(\varphi_1\,\mathbf{W}\,\varphi_2)]\!]$ is the set of states s_0 such that some infinite path $s_0 s_1 s_2 \cdots$ of the Kripke structure satisfies either $s_i \in [\![\varphi_1]\!]$ for every $i \in \mathbb{N}$, or $s_k \in [\![\varphi_2]\!]$ and $s_0, \ldots, s_{k-1} \in [\![\varphi_1]\!]$ for some $k \geq 1$. We extend the semantics to contexts and contextual formulas as for the μ-calculus.

We proceed to solve the validity problem for contextual CTL using the syntax-guided translation from CTL to the μ-calculus [6, Pag. 1066]. The translation assigns to each CTL-formula φ a closed formula φ^μ of the μ-calculus such that $[\![\varphi]\!] = [\![\varphi^\mu]\!]$ holds for every Kripke structure $\mathcal{K}$ and mapping ℓ.

▶ **Definition 32** (CTL to μ-calculus translation). *For any context or contextual CTL formula φ we inductively define the μ-calculus formula φ^μ by*

1. $p^\mu = p$
2. $(\neg p)^\mu = \neg p$
3. $([\,])^\mu = [\,]$
4. $(c[\psi])^\mu = c[\psi^\mu]$
5. $(\varphi_1 \wedge \varphi_2)^\mu = \varphi_1^\mu \wedge \varphi_2^\mu$
6. $(\varphi_1 \vee \varphi_2)^\mu = \varphi_1^\mu \vee \varphi_2^\mu$
7. $\mathbf{A}\,(\varphi_1 \,\mathbf{U}\, \varphi_2)^\mu = \mu X.([\cdot]\, X \wedge \varphi_1^\mu) \vee \varphi_2^\mu$.
8. $\mathbf{A}\,(\varphi_1 \,\mathbf{W}\, \varphi_2)^\mu = \nu X.([\cdot]\, X \wedge \varphi_1^\mu) \vee \varphi_2^\mu$.
9. $\mathbf{E}\,(\varphi_1 \,\mathbf{U}\, \varphi_2)^\mu = \mu X.(\langle\cdot\rangle\, X \wedge \varphi_1^\mu) \vee \varphi_2^\mu$.
10. $\mathbf{E}\,(\varphi_1 \,\mathbf{W}\, \varphi_2)^\mu = \nu X.(\langle\cdot\rangle\, X \wedge \varphi_1^\mu) \vee \varphi_2^\mu$.

The translations of **AF**, **AG**, **EF**, *and* **EG** *follow by instantiating (7-10) appropriately.*

The proof of the following corollary can be found in [8]. Intuitively, it is a consequence of the fact that the canonical instantiation of Definition 23 is a formula of the CTL-fragment of the μ-calculus.

▶ **Corollary 33.** *For any contextual CTL formula φ, let $\kappa_\varphi : C \to \mathrm{CTL}$ be the instantiation of contexts defined by*

$$\kappa_\varphi(c) := \bigwedge_{c[\psi] \in CSub(\varphi)} \big(\mathbf{AG}\,([\,] \to \psi^d)\big) \to p_{c[\psi]}$$

Then, the following statements are equivalent:

1. *φ is valid,*
2. *$\kappa_\varphi(\varphi)$ is valid, and*
3. *$\varphi_e := \Big(\bigwedge_{c[\psi_1], c[\psi_2] \in CSub(\varphi)} \mathbf{AG}\,(\mathbf{AG}\,(\psi_1^d \to \psi_2^d) \to (p_{c[\psi_1]} \to p_{c[\psi_2]}))\Big) \to \varphi^d$ is valid.*

Proof sketch (full proof in [8]). A straightforward induction shows that $(\sigma(\varphi))^\mu = \sigma^\mu(\varphi^\mu)$ for any instantiation σ. Moreover, $\kappa_\varphi^\mu = \kappa_{\varphi^\mu}$ and $\varphi_e^\mu = (\varphi^\mu)_e$. Hence, going back and forth between CTL and the μ-calculus, we can translate Theorems 26 and 30 to CTL. ◀

▶ **Corollary 34.** *The validity problem for contextual CTL formulas is* EXPTIME*-complete.*

Proof. The validity problem for CTL is known to be EXPTIME-complete [7], it is a specific case of the contextual validity problem, and Corollary 33 gives a polynomial reduction from the latter to the former. ◀

▶ **Example 35.** Let us use item (2) of Corollary 33 to show that the Boole-Shannon expansion is not valid in CTL. As in Example 11, let $\varphi := c[p] \leftrightarrow (p \wedge c[\mathrm{true}]) \vee (\neg p \wedge c[\mathrm{false}])$. By definition, $\kappa_\varphi(\varphi) := \kappa_\varphi(c[p]) \leftrightarrow \big((p \wedge \kappa_\varphi(c[\mathrm{true}])) \vee (\neg p \wedge \kappa_\varphi(c[\mathrm{false}]))\big)$. Simplification yields

$$\kappa_\varphi(\varphi) \quad \equiv \quad \begin{array}{c} p_{c[p]} \wedge p_{c[\mathrm{true}]} \wedge \big(\mathbf{AG}\,\neg p \to p_{c[\mathrm{false}]}\big) \\ \leftrightarrow \\ \big(p \wedge \big(\mathbf{AG}\, p \to p_{c[p]}\big) \wedge p_{c[\mathrm{true}]}\big) \vee \big(\neg p \wedge p_{c[p]} \wedge p_{c[\mathrm{true}]} \wedge p_{c[\mathrm{false}]}\big) \end{array}$$

This formula is not valid. For example, take any Kripke structure with a state s satisfying p and $p_{c[\text{true}]}$, but neither $p_{c[p]}$ nor $\mathbf{AG}\, p$. Then s satisfies the right-hand side of the bi-implication, because it satisfies the left disjunct, but not the left-hand side, because it does not satisfy $p_{c[p]}$. By Corollary 33, the Boole-Shannon expansion is not valid.

Either item (2) or (3) of Corollary 33 can be used to check, for instance, that the substitution rules $\mathbf{AG}\,(a \leftrightarrow b) \models \mathbf{AG}\,(c[a] \leftrightarrow c[b])$, and $\mathbf{AG}\,(a \to b) \models \mathbf{AG}\,(c[a] \to c[b])$ do hold.

4.5 Validity of contextual LTL formulas

The syntax of LTL is obtained by dropping $\mathbf{E}$ and $\mathbf{A}$ from the syntax of CTL. Formulas are interpreted over infinite sequences of atomic propositions [3], and a state s_0 of a Kripke structure satisfies a formula φ if every infinite path $s_0 s_1 s_2 \cdots$ of the Kripke structure satisfies φ. The Kripke structure itself satisfies φ if all its initial states satisfy φ.

Unlike for CTL, there is no syntax-guided translation from LTL to μ-calculus. However, there is one for *lassos*, finite Kripke structures in which every state has exactly one infinite path rooted at it.

▶ **Definition 36.** *A Kripke structure $\mathcal{K} = (S, \to, I, AP, \ell)$ is a* lasso *if S is finite and for every $s \in S$ there is exactly one state $s' \in S$ such that $s \to s'$.*

Consider the variation of the translation φ^μ from CTL where $\mathbf{X}$, $\mathbf{U}$, $\mathbf{G}$, and $\mathbf{F}$ are translated as $\mathbf{AX}$, $\mathbf{AU}$, $\mathbf{AG}$, and $\mathbf{AF}$. For example, we define $(\varphi_1\, \mathbf{U}\, \varphi_2)^\mu := \mu X.([\cdot]\, X \wedge \varphi_1^\mu) \vee \varphi_2^\mu$. We have:

▶ **Lemma 37.** *An LTL formula is valid iff it holds over all lassos. Further, for every lasso $\mathcal{K}$ and every formula φ of LTL, $\mathcal{K} \models_{LTL} \varphi$ iff $\mathcal{K} \models \varphi^\mu$.*

The results of Theorems 26 and 30 can be extended to LTL similarly to the CTL case. However, since φ^μ and φ are only guaranteed to be equivalent in lassos, some care should be taken to always use them.

▶ **Corollary 38.** *For any LTL formula φ, let $\kappa_\varphi : C \to \text{LTL}$ be the instantiation of contexts defined by*

$$\kappa_\varphi(c) := \bigwedge_{c[\psi] \in CSub(\varphi)} \big(\mathbf{G}\,([\;] \to \psi^d)\big) \to p_{c[\psi]}$$

Then, the following statements are equivalent

1. *φ is valid,*
2. *$\kappa_\varphi(\varphi)$ is valid, and*
3. *$\varphi_e := \Big(\bigwedge_{c[\psi_1], c[\psi_2] \in CSub(\varphi)} \mathbf{G}\,(\mathbf{G}\,(\psi_1^d \to \psi_2^d) \to (p_{c[\psi_1]} \to p_{c[\psi_2]}))\Big) \to \varphi^d$ is valid.*

Proof sketch (full proof in [8]). Like for CTL, $(\sigma(\varphi))^\mu = \sigma^\mu(\sigma^\mu)$, $\kappa_\varphi^\mu = \kappa_{\varphi^\mu}$, and $\varphi_e^\mu = (\varphi^\mu)_e$. Each implication of the equivalence can be derived as depicted in Figure 2. Notice that, while $\sigma(\varphi)$ and $\sigma^\mu(\varphi^\mu)$ are not equivalent in general, they are when evaluated on a lasso by Lemma 37. This allows going back and forth between LTL and the μ-calculus, and Lemma 25 and Proposition 29 complete the proof. ◀

Using the procedure just described, we can check that the rules in Figure 1 are valid, that the Boole-Shannon expansion does not hold in LTL, and one-side implications like $c[\mathbf{G}\, p] \models c[p]$, $\mathbf{G}\,(a \leftrightarrow b) \models \mathbf{G}\,(\varphi[a] \leftrightarrow \varphi[b])$ and $\mathbf{G}\,(a \to b) \models \mathbf{G}\,(\varphi[a] \to \varphi[b])$.

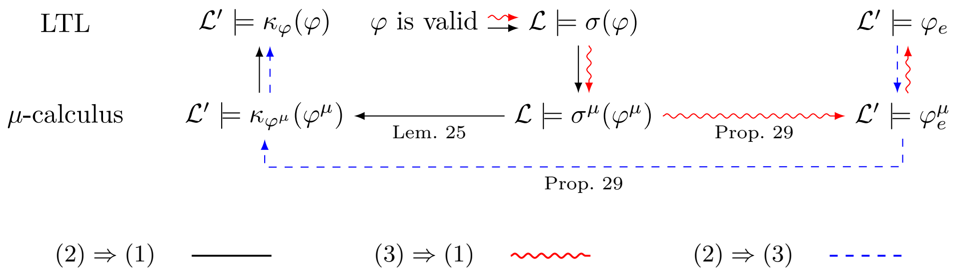

Figure 2 Proof summary of Corollary 38 (arrow is problem reduction).

5 Experiments

We have implemented the methods of Propositions 10 and 13 and Corollaries 33 and 38 in a prototype that takes two contextual formulas as input and tells whether they are equivalent, one implies another, or they are incomparable.[4] The prototype is written in Python and calls external tools for checking validity of ordinary formulas: MiniSat [5] through PySAT [10] for propositional logic, Spot 2.11 [4] for LTL, and CTL-SAT [11] for CTL. No tool has been found for deciding μ-calculus satisfiability, so this logic is currently not supported.

Most formulas of Figure 1 are solved in less than 10 milliseconds by the first and second methods. The hardest formula is the second one: using the equivalid formula, the largest automaton that appears in the process has 130 states and it is solved in 16.21 ms; with the canonical instantiation, the numbers are 36 and 20.22 ms. We have also applied small mutations to the rules of Figure 1 to yield other identities that may or may not hold (see Appendix C of [8] for a list). Solving them takes roughly the same time and memory as the original ones. For CTL, the behavior even with small formulas is much worse because of the worse performance of CTL-SAT. The canonical context method takes 20 minutes to solve $c[a \wedge b] \equiv c[a] \wedge c[b]$, while the method by the equivalid formula runs out of memory with that example and requires 22 minutes for the Boole-Shannon expansion. Hence, we have not continued with further benchmarks on CTL. The first three rows of Table 1 show the time, peak memory usage (in megabytes), and number of states of the automata (for LTL) required for checking the aforementioned examples. The experiments have been run under Linux in an Intel Xeon Silver 4216 machine limited to 8 Gb of RAM. Memory usage is as reported by Linux cgroups' memory controller.

In addition to these natural formulas, we have tried with some artificial ones with greater sizes and nested contexts to challenge the performance of the algorithm. We have considered two repetitive expansions of the rules in Figure 1:

1. There is a dual of the first rule in Figure 1 that removes a **W**-node below a **U**-node:

$$\varphi \,\mathbf{U}\, c[\psi_1 \,\mathbf{W}\, \psi_2] \equiv \varphi \,\mathbf{U}\, c[\psi_1 \,\mathbf{U}\, \psi_2] \vee (\mathbf{FG}\, \varphi \wedge (\varphi \wedge \mathbf{F}\, c[\mathrm{true}]) \,\mathbf{W}\, c[\psi_1 \,\mathbf{W}\, \psi_2]).$$

   Then, we can build formulas like $c_1[\psi_1 \,\mathbf{U}\, \psi_2] \,\mathbf{W}\, \varphi$ $(n = 1)$, $c_1[\psi_1 \,\mathbf{U}\, c_2[\psi_2 \,\mathbf{W}\, \psi_3]] \,\mathbf{W}\, \varphi$ $(n = 2)$, $c_1[\psi_1 \,\mathbf{U}\, c_2[\psi_2 \,\mathbf{W}\, c_3[\psi_3 \,\mathbf{U}\, \psi_4]]] \,\mathbf{W}\, \varphi$ $(n = 3)$, and so on, and apply the first rule of the rewrite system and its dual to obtain the normalized right-hand side. Table 1 shows that the first method does not finish within an hour for $n = 2$, and the second reaches this time limit for $n = 3$. For $n = 2$ the second method checks the emptiness of an automaton of 1560 states (and another of 720 states for the other side of the implication).

[4] The prototype and its source code are publicly available at `https://github.com/ningit/ctxform`.

Table 1 Compared performance with the challenging examples (memory in Mb).

Example		Method 1 Time	Method 1 Memory	Method 1 States	Method 2 Time	Method 2 Memory	Method 2 States
Shannon	Bool	8.18 ms	3.67		15.08 ms	3.67	
	LTL	5 ms	2.62	9	8.07 ms	2.62	12
Rules [9] (max)		23.88 ms	5.24	36	16.21 ms	4.71	130
Mutated (max)		44.12 ms	5.72	48	31.68 ms	4.98	130
(1)	0	1.66 ms	1.05	4	1.53 ms	1.05	4
	1	17.40 ms	4.92	36	15.62 ms	4.46	130
	2	timeout			3:25 min	413.83	1560
(2)	1	36.19 ms	5.24	45	22.40 ms	5.24	80
	2	623.07 ms	26.96	168	117.79 ms	10.49	160
	3	5:21 min	1245.95	1140	26.04 s	100.25	220

2. The third and fourth rules of Figure 1 can also be nested. We can consider $c_0[\mathbf{FG}\, c_1[p]]$ $(n = 1)$, $c_0[\mathbf{FG}\, c_1[\mathbf{GF}\, c_2[p]]]$ $(n = 2)$, $c_0[\mathbf{FG}\, c_1[\mathbf{GF}\, c_2[\mathbf{FG}\, c_3[p]]]]$ $(n = 3)$, and so on. We also take $c_0 = c_1 = \cdots = c_n$ to make the problem harder. As shown in Table 1, we can solve up to $n = 3$ within the memory constraints.

6 Conclusions

We have presented two different methods to decide the validity and satisfiability of contextual formulas in propositional logic, LTL, CTL, and the μ-calculus. Moreover, we have shown that these problems have the same complexity for contextual and ordinary formulas. Interesting contextual equivalences can now be checked automatically. In particular, we have replaced the manual proofs of the several LTL simplification rules in [9] to a few milliseconds of automated check.

While we have limited our exposition to formulas in negation normal form, and hence to monotonic contexts, the results for propositional logic, CTL, and LTL can be generalized to the unrestricted syntax of the corresponding logics and to arbitrary contexts. Some clues are given in Appendix B of [8].

References

1 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021. doi:10.3233/FAIA336.

2 Julian C. Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer, 2018. doi:10.1007/978-3-319-10575-8_26.

3 Edmund M. Clarke, Thomas A. Henzinger, and Helmut Veith. Introduction to model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 1–26. Springer, 2018. doi:10.1007/978-3-319-10575-8_1.

4 Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. From Spot 2.0 to Spot 2.10: What's new? In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2022. doi:10.1007/978-3-031-13188-2_9.

5 Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003. doi:10.1007/978-3-540-24605-3_37.

6 E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 995–1072. Elsevier and MIT Press, 1990. doi:10.1016/B978-0-444-88074-1.50021-4.

7 E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1):1–24, 1985. doi:10.1016/0022-0000(85)90001-7.

8 Javier Esparza and Rubén Rubio. Validity of contextual formulas (extended version). *arXiv*, 2024. doi:10.48550/arXiv.2407.07759.

9 Javier Esparza, Rubén Rubio, and Salomon Sickert. Efficient normalization of linear temporal logic. *J. ACM*, 71(2):16:1–16:42, 2024. doi:10.1145/3651152.

10 Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

11 Nicola Prezza. CTL (Computation Tree Logic) SAT solver, 2014. URL: https://github.com/nicolaprezza/CTLSAT.

12 Rubén Rubio. Equivalence checker for contextual formulas. Software (visited on 2024-07-25). URL: https://github.com/ningit/ctxform.