

Electronic Notes in Theoretical Computer Science 223 (2008) 201–213

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

The Complexity of Satisfiability for Fragments of CTL and CTL*\** [1](#_bookmark2)

Arne Meier,[a](#_bookmark0)*,*[2](#_bookmark2) Martin Mundhenk,[b](#_bookmark1)*,*[3](#_bookmark2) Michael Thomas[a](#_bookmark0)*,*[4](#_bookmark2) and Heribert Vollmer[a](#_bookmark0)*,*[5](#_bookmark2)

a *Theoretische Informatik Gottfried Wilhelm Leibniz Universita¨t* *Appelstr. 4, 30167 Hannover, Germany*

b *Institut fu¨r Informatik Friedrich-Schiller-Universita¨t 07737 Jena, Germany*

**Abstract**

The satisfiability problems for CTL and CTL*s* are known to be EXPTIME-complete, resp. 2EXPTIME- complete (Fischer and Ladner (1979), Vardi and Stockmeyer (1985)). For fragments that use less temporal or propositional operators, the complexity may decrease. This paper undertakes a systematic study of satisfiability for CTL- and CTL*s*-formulae over restricted sets of propositional and temporal operators. We show that restricting the temporal operators yields satisfiability problems complete for 2EXPTIME, EXPTIME, PSPACE, and NP. Restricting the propositional operators either does not change the complexity

(as determined by the temporal operators), or yields very low complexity like NC1, TC0, or NLOGTIME.

*Keywords:* Temporal Logic, Satisfiability, Post’s Lattice.

# Introduction

For reasoning about the ongoing behaviour of programs, in particular non-terminating programs such as operating systems, the branching time logic CTL٨, introduced by Emerson and Halpern [[7](#_bookmark30)] (see also [[6](#_bookmark31)]), has been advocated to be a good language [[19](#_bookmark44)], and in the meantime it has proven to be useful even for practical purposes.

A decidable satisfiability problem is central for such logics in order to be a useful tool in program verification. For CTL٨, satisfiability was proven to be complete

1 Supported in part by DFG VO 630/6-1.

2 Email: [meier@thi.uni-hannover.de](mailto:meier@thi.uni-hannover.de)

3 Email: [mundhenk@cs.uni-jena.de](mailto:mundhenk@cs.uni-jena.de)

4 Email: [thomas@thi.uni-hannover.de](mailto:thomas@thi.uni-hannover.de)

5 Email: [vollmer@thi.uni-hannover.de](mailto:thomas@thi.uni-hannover.de)

1571-0661 © 2008 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2008.12.040

for double exponential time by Vardi and Stockmeyer [[19](#_bookmark44)]. For certain fragments, satisfiability is known to be more efficiently decidable: Sistla and Clarke [[18](#_bookmark43)] proved that for linear temporal logic LTL, the fragment of CTL٨ not allowing path quantifiers, the satisfiability problem is complete for polynomial space. For the fragment of LTL that disallows U, satisfiability is NP-complete. Markey [[12](#_bookmark37)] extended these results showing essentially that adding operators for the past does not increase complexity (“past is for free”). Further fragments of LTL were classified in [[2](#_bookmark26)]. Fischer and Ladner [[8](#_bookmark32)] proved that for computation tree logic CTL, the fragment of CTL٨ in which each path quantifier is followed by exactly one temporal operator that is not a path quantifier (i. e., X, U, F, G), satisfiability is complete for exponential time. However, a systematic study of the complexity of the satisfiability problem for fragments of CTL٨ has not been undertaken until today. This is the purpose of the present paper.

We first consider fragments of CTL and of CTL٨ where we restrict the allowed temporal operators. Here, a CTL-operator is a pair of a path quantifier (A and E) and non-path operator (X, U, etc.). We determine the lattice of all sets of temporal operators where one such set *T*1 is below another set *T*2 (*T*1 *± T*2) iff the operators from *T*1 can be expressed using operators *T*2. Then we determine *for each set in the lattice* the complexity of the satisfiability problem restricted to only these temporal operators.

For CTL, we show, e. g., that the complexity of the satisfiability problem drops to NP-complete for the operators sets *∅* and *{*AF*}*, it is PSPACE-complete for *{*AX*}*,

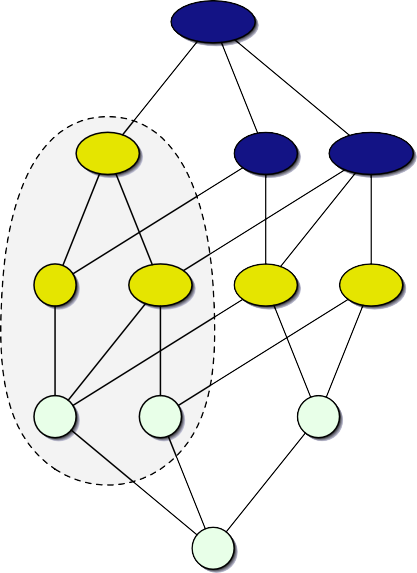
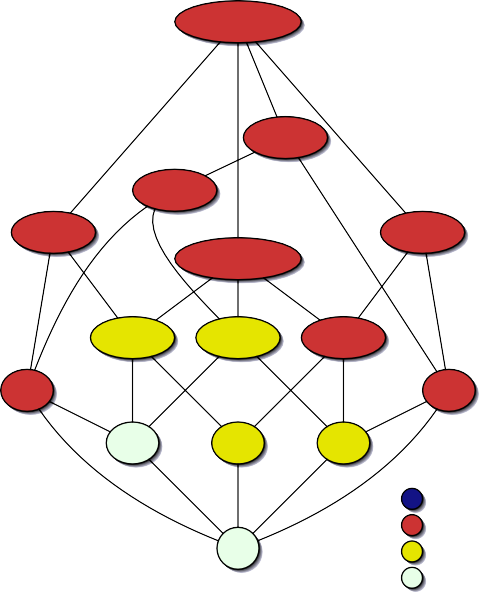
*{*AG*}*, *{*AX*,* AF*}*, *{*AF*,* AG*}*, and is complete for exponential time for all other cases. For CTL٨, we show, e. g., that the complexity of the satisfiability problem drops to NP-complete for the operators sets *∅*, *{*A*}*, *{*F*}*, and *{*X*}*, it is PSPACE-complete for

*{*U*}*, *{*X*,* F*}* and *{*U*,* X*}*, *{*A*,* F*}* and *{*A*,* X*}*, and is complete for double exponential

time for all other cases. Figure [1](#_bookmark3) summarizes these results.

As a second step, we also restrict the allowed propositional operators, following the approach undertaken in [[2](#_bookmark26)] for LTL. Let CTL٨-SAT(*T, B*) denote the satisfiability problem for CTL٨ restricted to the fragments of formulae only allowing temporal operators from *T* and propositional operators from *B*. Here, we thus have to consider the lattice of all classes of Boolean functions, and we say that for such classes *B*1*, B*2, *B*1 *± B*2 if all functions in *B*1 can be obtained by superposition (essentially simple composition or substitution of functions) from functions in *B*2. This lattice is the well known *Post’s lattice* (cf., e. g., [[14](#_bookmark39),[3](#_bookmark28)]), see Fig. [2](#_bookmark4).

It turns out that if *B* contains (or can implement) the negation of implication *x /→ y* (that is, *x ∧ ¬y*)—in terms of Post’s lattice this means that S1 *⊆ B*— then satisfiability is as complicated as if allowing all propositional operators or a complete set such as *{∧, ¬}*, in other words, the complexity of CTL٨-SAT(*T, B*) is determined by the set *T* as described above (and independent of the actual *B*). If on the other hand *B* cannot implement the negation of implication, then the complexity of CTL٨-SAT(*T, B*) drops to a very low class inside the circuit class NC1. In this case, the complexity of CTL٨-SAT(*T, B*) astonishingly is independent of the temporal operators we allow. For example, if we consider only monotone formulae,

AX*,* AF*,* EU A*,* X*,* U

AG*,* AU

AF*,* EU

LTL

X*,* U

A*,* U A*,* X*,* F

AX*,* AU

AX*,* AF*,* AG

AX*,* EU

U X*,* F

A*,* F

A*,* X

AX*,* AF AF*,* AG AX*,* AG

AU EU

F X A

AF AX AG

2EXPTIME-complete EXPTIME-complete

*∅* PSPACE-complete *∅*

NP-complete

Fig. 1. The lattice of CTL-operators (left) and CTL*s*-operators (right). Nodes are labelled with a minimal set of operators; colors indicate the complexity of the satisfiability problem without restrictions on the Boolean connectives.

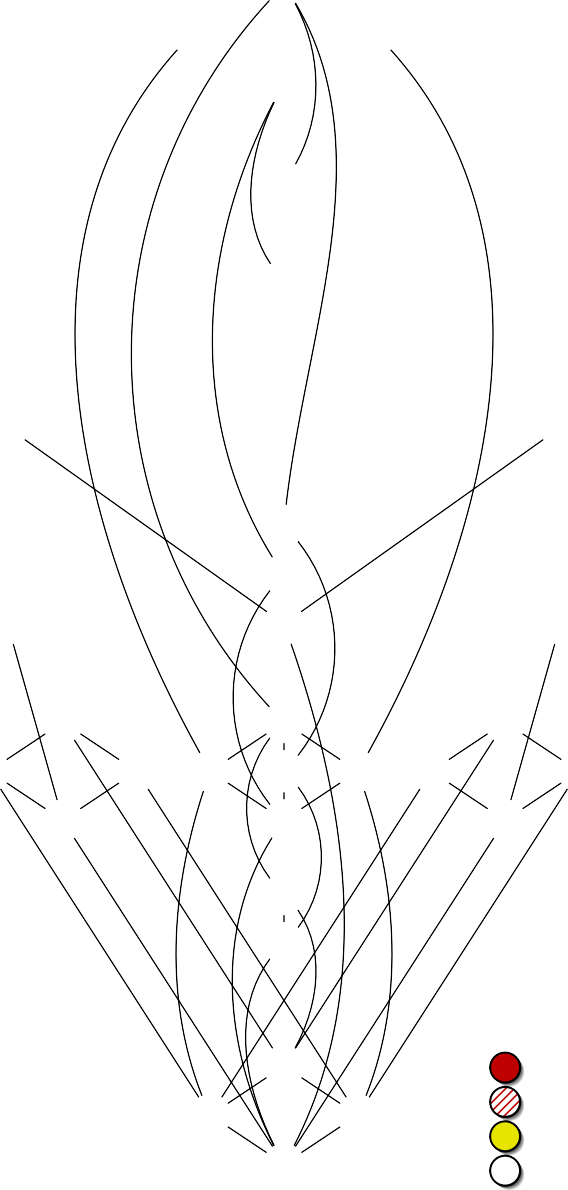
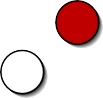
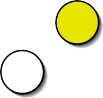
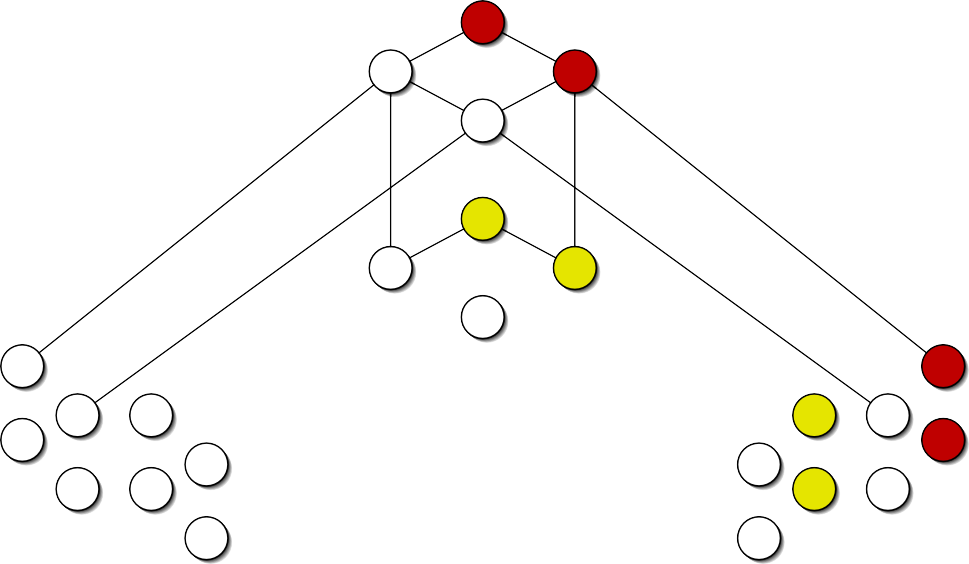
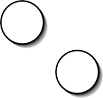
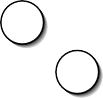
i. e., *B* = *{∧, ∨,* true*,* false*}* (this corresponds to the class M in Post’s lattice), then CTL٨-SAT(*T, B*) is complete for NC1 for all *T* . If *B ⊆ {∧,* true*,* false*}* (*B ⊆* E in Post’s lattice) then CTL٨-SAT(*T, B*) is complete for TC0 for all *T* . These results for the case of unrestricted temporal operators are summarized in Figure [2](#_bookmark4). It should be remarked that also in the case of simple propositional satisfiability, the operator *x ∧ ¬y* determines if the problem is NP-complete or in P, see [[11](#_bookmark36)].

In this vein, we study the complexity of satisfiability for CTL and for CTL٨ for all combinations of *B* and *T* . We give completeness results for 2EXPTIME, EXPTIME, PSPACE, NP, NC1, and TC0. However, we have to leave open one scenario: When *B* consists only of the exclusive-or (plus possibly the constants *true* and *false*) we can only state a trivial upper bound. We come back to this open case in the conclusion.

The reader might expect certain fragments of CTL٨ to have a trivial satisfia- bility problem (e. g., since the allowed formulas are always satisfiable)—the lowest complexity in our classification, however, is completeness for TC0. The reason is that the syntax alone, checking that a given word is a correct formula, leads to TC0-completeness. In order to determine the cases of trivial satisfiability we therefore also study the *promise problem* to determine, given a syntactically correct formula as input, if it is satisfiable.

The rationale behind our approach is that looking for simpler fragments helps us to understand where the boundary lies between hard and easy fragments. This provides insight into the sources of hardness (*x ∧ ¬y* on the propositional side, and for instance U on the temporal side). We also hope that our results might lead to improved algorithms for the special cases. One of our technically most involved results concerns the CTL-satisfiability for the operators *{*AF*,* AG*}*: Here we consider quasi-models (models whose labels are certain variants of Hintikka sets) and prove

BF



R1 R0

R2

2

S

0

2 2

S

S

02 01

S3

M

M1 M0

M2

2

S

1

2 2

S

S

11 12

S3

0 S2 S2 1

3 3 00

S

S

02 01

3

S

00

S0 D

10 3 3

11 12

S

S

3

S

10

S1

S02 S01

S00

D1

D2 S10

S11

S12

V

V1 V0

V2

Complexity results for CTL: EXPTIME-complete in EXPTIME

NC1-complete TC0-complete

L

L1 L3 L0 L2

N N2

I

I1 I0

I2

E

E1 E0

E2

Complexity results for CTL٨: 2EXPTIME-complete in 2EXPTIME

NC1-complete TC0-complete

Fig. 2. Post’s lattice. Colors indicate the complexity of CTL-SAT and CTL*s*-SAT without restrictions on the temporal operators.

that a given formula *ϕ* is satisfiable iff a quasi-model with certain properties exists. A PSPACE-upper bound then is obtained by solving a certain reachability problem in the graph of quasi-models of *ϕ*. A clever implementation of this algorithm (and algorithms for other special cases) could lead to better tools than we have today.

The rest of this paper is organized as follows. Section [2](#_bookmark5) contains preliminaries. The complexity of the satisfiability problem for the computational tree logic, CTL, is considered in Section [3](#_bookmark7). Section [4](#_bookmark21) then extends these considerations to the logic

CTL٨. Section [5](#_bookmark25) concludes with a summary and a discussion.

In the interests of space, proofs are omitted or sketched. Detailed proof will be included in the full version of this paper.

# Preliminaries

* 1. *Complexity Theory*

We assume familiarity with the standard notions of complexity theory. In par- ticular, we will make use of the classes LOGSPACE*,* P*,* NP*,* PSPACE*,* EXPTIME and 2EXPTIME.

We require subtle reductions in order to obtain hardness results for complexity classes below TC0. Therefore, we introduce the following notion of reducibility (see [[21](#_bookmark46)]). Let *A* and *B* be languages. Then *A* is *constant-depth reducible* to *B* (*A ≤*cd *B*) if there exists an AC0-circuit family *{C*n*}*n>0 with *{л, V, ч}*-gates and oracle gates for *B* such that for all *x*, *C|*x*|*(*x*) = 1 iff *x ∈ A*. One of our results even addresses complexity issues inside the class AC0—hence *≤*cd-reducibility is of no use since AC0 forms the **0**-degree of *≤*cd. Instead, we will make use of *dlt-projection reducibility*

(*A ≤*

dlt proj

*B*) as introduced in [[16](#_bookmark41)]. We note that TC0 and NC1 are closed under *≤*cd,

and NLOGTIME and coNLOGTIME are closed under *≤*dlt .

proj

* 1. *Temporal Logic*

We inductively define CTL٨-formulae as follows. Let Φ be a finite set of atomic propositions. The symbols used are the atomic propositions in Φ, the constant symbols *T* and *⊥*, the Boolean connectives *ч* and *л*, the temporal operator symbols

X, U, and A. A is also called a *path quantiﬁer*, temporal operators aside from A

are thence also called *pure temporal operators*. The atomic propositions, *T* and *⊥* are called *atomic formulae*. Each atomic formula is a *state formula*, and each state formula is a *path formula*. Let *ϕ, ψ* be state formulae and *χ, π* be path formulae. Then (*ϕ*), *ϕ л ψ*, *чϕ*, A*χ* are state formulae, and *χ л π*, *чχ*, X*χ*, and [*χ*U*π*] are path formulae. The set of CTL٨-formulae (or *formulae*) is the union of all state formulae and of all path formulae. We further define CTL٨(*T, B*) to be the set of CTL٨-formulae using the Boolean connectives in *B* and the path quantifiers and

temporal operators in *T* only. The set of proper subformulae of *ϕ* will be denoted by SF(*ϕ*), the number of pure temporal operators in *ϕ* by #T(*ϕ*).

A *model* is a triple *M* = (*S, R, l*), where *S* is a finite set of states, *R ⊆ S×S* a total relation (i. e., for each *s ∈ S*, there exists an *s'* such that (*s, s'*) *∈ R*), and *l* : *S →* P(Φ) is a labelling function. A *path x* is an infinite sequence *x* = (*x*1*, x*2*,.. .*) *∈ S*ω such that (*x*i*, x*i+1) *∈ R*, for all *i >* 0.

Let *M* = (*S, R, l*) be a model, *χ* be a state formula, *s ∈ S* be a state and *x* = (*x*1*, x*2*,...* ) *∈ S*ω be a path. The truth of a CTL٨-formula w. r. t. *M* is inductively defined using the following semantics. Let *ϕ, ψ, χ, π ∈* CTL٨.

*M, s |*= *T* always,

*M, s |*= *⊥* never,

*M, s |*= *p* iff *p ∈* Φ and *p ∈ l*(*s*), *M, s |*= (*ϕ*) iff *M, s |*= *ϕ*,

*M, s |*= *чϕ* iff *M, s |*= *ϕ*,

*M, s |*= *ϕ л ψ* iff *M, s |*= *ϕ* and *M, s |*= *ψ*,

*M, s |*= A*χ* iff for all paths *x* = (*x*1*, x*2*,...* ) with *x*1 = *s* holds *M, x |*= *χ*, *M, x |*= *χ* iff *M, x*1 *|*= *χ*,

*M, x |*= X*χ* iff *M, x*2 *|*= *χ*,

*M, x |*= [*χ*U*π*] iff *M, x*k *|*= *π*, for some *k ∈* N, and *M, x*i *|*= *χ*, for all 1 *≤ i < k*.

The syntax and semantics of each remaining Boolean function *f* can be expressed through the connectives *л* and *ч*. The remaining temporal operators are defined in the following way:

E*ϕ ≡ ч*A*чϕ,* F*ϕ ≡ T*U*ϕ,* G*ϕ ≡ ч*F*чϕ,*

where E is again also called a path quantifier. A state formula *ϕ* is hence said to be *satisﬁed by model M* if there exists an *s ∈ S* such that *M, s |*= *ϕ* (written as *M |*= *ϕ*). Analogously, a path formula is said to be satisfied by a model *M* if there exists a path *x* = (*x*1*, x*2*,.. .*) such that *M, x |*= *ϕ*. Finally *ϕ* is said to be *satisﬁable* if there exists a model *M* that satisfies *ϕ*. We define CTL٨-SAT(*T, B*) to be the problem of deciding whether a given CTL٨(*T, B*)-formula is satisfiable.

A CTL*-formula* is a CTL٨-formula in which each path quantifier is followed by exactly one pure temporal operator and each pure temporal operator is preceded by exactly one path quantifier. The set of CTL-formulae is a strict subset of the set of CTL٨-formulae. For example, AGEF*p* is a CTL-formula, whereas A(GF*p →* F*q*) is not. Pairs of path quantifiers and pure temporal operators are thence also referred to as CTL*-operators*. Let ALL denote the set of all CTL-operators. We remark the following dualities among CTL-operators:

EX*ϕ ≡ ч*AX*чϕ,* EF*ϕ ≡* E[*T*U*ϕ*]*,* AF*ϕ ≡* A[*T*U*ϕ*]*,* AG*ϕ ≡ ч*EF*чϕ,* EG*ϕ ≡ ч*AF*чϕ,*

and A[*ψ*U*χ*] *≡* AF*χ л ч*E[*чχ*U(*чψ л чχ*)]. Hence *{*AX*,* AF*,* EU*}* is a minimal set of CTL-operators for CTL (in presence of all Boolean connectives), whereas *{*AX*,* AG*,* AU*}* is not complete for CTL [[10](#_bookmark35)]. Alike CTL٨-SAT, we define CTL(*T, B*) to be the set of all CTL-formulae using the CTL-operators in *T* and the Boolean connectives in *B* only, and define CTL-SAT(*T, B*) to be the problem of deciding whether a given CTL(*T, B*)-formula is satisfiable.

* 1. *Boolean Clones*

Since there are infinitely many finite sets *B* of Boolean functions, we introduce some algebraic tools to classify the complexity of the infinitely many arising satisfiability problems. A set *B* of Boolean functions is called a *clone* if it is closed under superposition, which means *B* contains all projections and *B* is closed under arbitrary composition [[14](#_bookmark39), Chapter 1]. For a set *B* of Boolean functions we denote with [*B*] the smallest clone containing *B* and call *B* a base for [*B*]. In [[15](#_bookmark40)] Post classified the

lattice of all clones and found a finite base for each clone, see Fig. [2](#_bookmark4). In order to introduce the clones relevant to this paper, we define the following notions, where *f* is an *n*-ary Boolean function.

* + - *f* is 1*-reproducing* if *f* (1*,...,* 1) = 1.
    - *f* is *monotone* if *a*1 *≤ b*1*, a*2 *≤ b*2*,..., a*n *≤ b*n =*⇒ f* (*a*1*,..., a*n) *≤ f* (*b*1*,..., b*n).
    - *f* is 1*-separating* if there exists an *i ∈ {*1*,..., n}* such that *f* (*a*1*,..., a*n) = 1 implies *a*i = 1.
    - *f* is *self-dual* if *f ≡* dual(*f* ), where dual(*f* )(*x*1*,..., x*n) = *чf* (*чx*1*,..., чx*n).
    - *f* is *linear* if *f ≡ x*1 *⊕· · ·⊕ x*n *⊕ c* for a constant *c ∈* 0*,* 1 and variables *x*1*,..., x*n.

The clones relevant to this paper are listed in Table [1](#_bookmark6). The definition of all Boolean clones can be found, e. g., in [[3](#_bookmark28)].

Name BF R1 M

S1

D L

Definition

All Boolean functions

*{f* : *f* is 1-reproducing*}*

*{f* : *f* is monotone*}*

*{f* : *f* is 1-separating*}*

*{f* : *f* is self-dual*}*

*{f* : *f* is linear*}*

Base

*{л, ч}*

*{V, →}*

*{V, л, ⊥, T}*

*{x л y}*

*{xy V xz V* (*y л z*)*}*

*{⊕, ⊥}*

V *{f* : *f ≡ c*0 *V* n

i=1

V0 [*{V}*] *∪* [*{⊥}*]

E *{f* : *f ≡ c*0 *л* n

i=1

E0 [*{л}*] *∪* [*{⊥}*]

*c*i*x*i where the *c*is are constant*}*

*c*i*x*i where the *c*is are constant*}*

*{V, ⊥, T}*

*{V, ⊥}*

*{л, ⊥, T}*

*{л, ⊥}*

N *{f* : *f* depends on at most one variable*}*

I *{f* : *f* is a projection or a constant*}*

Table 1

A list of Boolean clones with definitions and bases.

*{ч, ⊥, T}*

*{*id*, ⊥, T}*

# Satisfiability in CTL

We commence by considering the complexity of the satisfiability problem for restricted sets of the CTL-operators and continue with restricted sets of Boolean functions. Recall the previously known results.

**Theorem 3.1 ([**[**5**](#_bookmark33)**], [**[**8**](#_bookmark32)**])** (i) CTL*-*SAT(*∅,* BF) *is* NP*-complete.*

(ii) CTL*-*SAT(*{*AX*,* AU*,* EU*},* BF) *is* EXPTIME*-complete.*

* 1. *Restricting the CTL-operators*

**Theorem 3.2** *Let T be a set of* CTL*-operators. Then* CTL*-*SAT(*T,* BF) *is*

1. NP*-complete under ≤*cd*-reductions if T* = *{*AF*},*
2. PSPACE*-complete under ≤*cd*-reductions if T* = *{*AG*},{*AX*},{*AF*,*AG*},{*AX*,*AF*},*
3. EXPTIME*-complete under ≤*cd*-reductions in all other cases.*

**Proof (Sketch)** For ([i](#_bookmark9)), NP-hardness of CTL-SAT(*{*AF*,* BF*}*) is immediate from Theorem [3.1](#_bookmark8)([i](#_bookmark9)). The membership in NP follows from a small model property: *ϕ ∈* CTL(*{*AF*,* BF*}*) is satisfiable iff *ϕ* is satisfiable in a model of size *≤ |ϕ|*O(1). As the model-checking problem for CTL is polynomial-time solvable [[4](#_bookmark29)], we can hence simply guess a model *M* and check whether *M |*= *ϕ*.

As for ([ii](#_bookmark10)), it suffices to show PSPACE-hardness for *T* = *{*AG*}, {*AX*}*, and membership in PSPACE for *T* = *{*AF*,* AG*}, {*AX*,* AF*}*. The hardness for both *T* =

*{*AG*}*, and *T* = *{*AX*}* is established using similar *≤*cd-reductions *f*AX and *f*AG from the satisfiability problem for quantified Boolean formulae, QBF-SAT. For *ϕ ∈* QBF-SAT, the constructed formula *f* (*ϕ*) forces any satisfying model to encode

in a tree-like structure the set of assignments necessary to fulfill *ϕ*. Both proofs are similar to [[9](#_bookmark34), Theorem 3.1]; a proof for CTL-SAT(*{*AX*},* BF) *∈* PSPACE can also be found in [[13](#_bookmark38), Theorem 9].

Now consider *T* = *{*AF*,* AG*}*. To show memberhip in PSPACE, we present an algorithm inspired by the algorithm showing that provability in the modal logic K is in PSPACE [[9](#_bookmark34)]. The algorithm is based on the notion of *quasi models*: let *ϕ ∈* CTL(*{*AF*,* AG*},* BF) be in negation normal form (i. e., negations occur in front of atomic formulae only), a quasi model for *ϕ* is a model *M* = (*S, R, l*) with labels *l* : *S →* P(CTL(*{*AF*,* AG*},* BF)) such that

1. for all *s ∈ S*, *l*(*s*) are minimal sets satisfying the condition that *ψ л χ ∈ l*(*s*) implies *ψ ∈ l*(*s*) and *χ ∈ l*(*s*), and the condition that *ψ V χ ∈ l*(*s*) implies *ψ ∈ l*(*s*) or *χ ∈ l*(*s*),
2. *ϕ ∈ l*(*s*) for some *s ∈ S*.
3. for all *s ∈ S* and each *Oψ ∈ l*(*s*), “*M* satisfies the constraints imposed by *Oψ*”,

i. e., *O ∈ {*AF*,* EF*,* AG*,* EG*}*, *ψ ∈ l*(*x*i) on all/some paths *x* = (*x*1*, x*2*, x*3*,.. .*), *x*1 = *s*, and all/some 1 *≤ i ∈* N.

Note that the labels of quasi models bear resemblance to Hintikka sets but differ from the latter in that they are allowed to contain *⊥*. The algorithm is based on the following observation: *ϕ* is satisfiable iff there is a quasi-model for *ϕ* whose labels are consistent on all path prefixes of linear length from some *s ∈ S* with *ϕ ∈ l*(*s*). Thence the algorithm performs a nondeterministic depth-first search for contradictions on the set of quasi models for *ϕ*. The space-bound derives from the linear length of path prefixes to be investigated.

For *T* = *{*AX*,* AF*}*, a straightforward modification of the former algorithm is not possible, since the X operator allows for the construction of “counters” such that contradictions may occur in exponential depth firstly. Nevertheless,

any CTL(*{*AX*,* AF*},* BF)-formula may impose at most linearly many temporal con- straints. Using the fixpoint-characterisation of EG, we derive an algorithm for *ϕ ∈* CTL(*{*AX*,* AF*},* BF) in a two-steps approach: first verify that *ϕ* with all EG operators ignored is satisfiable, then test each of the EG-prefixed subformulae for satisfiability separately.

Finally for ([iii](#_bookmark11)), membership in EXPTIME is due to Theorem [3.1](#_bookmark8)([iii](#_bookmark11)). Hardness for EXPTIME is obtained from reducing the word problem for polynomial-space alternating Turing machines to CTL-SAT(*T,* BF). The reduction for *T* = *{*AX*,* AG*}* is straightforward and can then be modified to prove hardness for the cases *T* = *{*AU*}* and *T* = *{*EU*}*, too. The hardness of the remaining fragments follows.

* 1. *Restricting the Boolean connectives*

Say that a set *B* of Boolean connectives is *non-trivial* if *B* contains a connective of arity *≥* 2. We state an auxiliary lemma.

**Lemma 3.3** *Let B be a non-trivial set of Boolean function symbols and let T be a* *set of* CTL*-operators. The problem to decide, whether a given string is a* CTL(*T, B*)*- formula, is complete for* TC0 *under ≤*cd*-reductions.*

**Theorem 3.4** *Let T denote a set of* CTL*-operators and let B be a set of Boolean functions such that* [*B*] *∈/ {*L*,* L0*} and B is non-trivial. Then* CTL*-*SAT(*T, B*) *is*

1. *equivalent to* CTL*-*SAT(*T,* BF) *if* S1 *⊆* [*B*]*,*
2. NC1*-complete under ≤*cd*-reductions if* S11 *⊆* [*B*] *⊆* M*, and*
3. TC0*-complete under ≤*cd*-reductions in all other cases.*

**Proof (Sketch)** For ([i](#_bookmark14)), note that BF = [S1 *∪ {T}*] = [*B ∪ {T}*]. It hence suffices to show that we can generate the constant *T* in all sets of Boolean functions *B* satisfying [*B*] = S1.

For ([ii](#_bookmark15)), [*B*] does not contain negations. Hence we can substitute each atomic

proposition with *T* and evaluate this proposition-free formula alike propositional formulae [[17](#_bookmark42), Theorem 3.2]. As evaluation of propositional S11-formula is NC1- complete already, the claim follows.

For ([iii](#_bookmark16)), we have to distinguish between two cases. First consider the cases

[*B*] *⊆* R1 and [*B*] *⊆* D. An induction on the formulae structure shows that all formulae are trivially satisfiable by the model *M* = (*{s}, {*(*s, s*)*}, l*), where, for all *s ∈ S*, either *l*(*s*) = Φ or *l*(*s*) = *∅*. If [*B*] *⊆* N, we can w. l. o. g. transform the given formula *ϕ* to be of the form

*ϕ ≡ 0*1*0*2 *··· 0*k*У*1 *ψ*U*У*2 *···* U*У*l[*···* U*ψ'*] *···* *,*

where *ψ ∈* CTL(*T, B*), *ψ' ∈* CTL(*T \ {*AU*,* EU*}, B*), *0*1*,..., 0*k *∈ T \ {*AU*,* EU*}* and

*У*1*,..., У*l *∈ {*A*,* E*}*. Hence we only need to count the number of preceding negations of *ψ'* modulo 2. For the remaining clones we can substitute the propositions with

*T* and only need to search for a *T* (in the *V*-case), or ensure absence of *⊥* (in the

*л*-case). Having established membership in TC0, completeness for TC0 stems from Lemma [3.3](#_bookmark12).

That is, for the last case of Theorem [3.4](#_bookmark13), the main complexity thus lies in checking the syntactical correctness of the given formula. In order to classify the complexity of CTL-SAT(*T, B*) beyond the complexity of its syntactical correctness, we restrict our attention to syntactically correct input formulae: Let CTL-SAT*P* (*T, B*) denote the promise problem of deciding whether a given syntactically correct CTL(*T, B*)-formula is satisfiable. The following theorem refines Theorem [3.4](#_bookmark13) for subclasses of TC0.

**Theorem 3.5** *Let T denote a set of* CTL*-operators and let B be a set of Boolean* *functions such that* CTL*-*SAT(*T, B*) *is* TC0*-complete. Then* CTL*-*SAT*P* (*T, B*) *is*

1. *in* TC0 *if T ∩ {*AU*,* EU*} /*= *∅ and* [*B*] *∈ {*V*,* V0*,* E*,* E0*,* N*},*
2. NLOGTIME*-complete under ≤*dlt *-reductions if T ∩ {*AU*,* EU*}* = *∅ and* [*B*] *∈*

proj

*{*V*,* V0*},*

1. coNLOGTIME*-complete under ≤*dlt

proj

*-reductions if T ∩ {*AU*,* EU*}* = *∅ and* [*B*] *∈*

*{*E*,* E0*},*

1. *equivalent to* MOD2 *under ≤*dlt

proj

*-reductions if T ∩ {*AU*,* EU*}* = *∅ and* [*B*] = N*,*

*and*

1. *trivial in all other cases.*

**Proof (Sketch)** For ([i](#_bookmark17)), one has to determine the relevant parts of the formula first. This requires counting the parentheses, therefore the problem remains in TC0.

The cases ([ii](#_bookmark18)) and ([iii](#_bookmark19)) can be solved analogously to [[17](#_bookmark42), Lemma 3.7], that is, by guessing the position of a satisfying *T* (or a falsifying *⊥*, resp.) after substituting all propositions with *T*. Hardness is obtained via a reduction from the language

*{*0*,* 1*}*٨1*{*0*,* 1*}* (or *{*0*,* 1*}*٨, resp.).

For ([iv](#_bookmark20)), syntactically correct formulae in CTL(*T,* N) can be checked for satisfia- bility by just counting the preceding negations modulo 2, for all temporal operators and Boolean connectives are unary. Hardness for this case arises from a reduction from PARITY = *{w ∈ {*0*,* 1*}*٨ *| |w|*1 *≡* 1 mod 2*}*.

Lastly, for any other combination of *T* and *B*, all CTL(*T, B*)-formulae are trivially satisfiable.

# Satisfiability in CTL

Having classified the complexity of the satisfiability problem for CTL, we now turn to CTL٨, a logic strictly more expressive than CTL: instead of paired, path quantifiers and temporal operators may occur independently of each other. This fact amounts to a jump in the complexity in the general case.

**Theorem 4.1 ([**[**19**](#_bookmark44)**])** CTL٨*-*SAT(*{*A*,* X*,* U*},* BF) *is* 2EXPTIME*-complete.*

We will hence proceed analogously to Section [3](#_bookmark7) and consider the complexity of the satisfiability problem for restricted sets of path quantifiers and temporal

operators and restricted sets of Boolean functions.

* 1. *Restricting the temporal operators and path quantiﬁers*

**Theorem 4.2** *Let T be a set of temporal operators. Then* CTL٨*-*SAT(*T,* BF) *is*

1. NP*-complete under ≤*cd*-reductions if T* = *∅, {*A*}, {*F*}, {*X*},*
2. PSPACE*-complete under ≤*cd*-reductions if T* = *{*U*},{*X*,*F*}, {*X*,*U*},{*A*,*X*},{*A*,*F*},*
3. 2EXPTIME*-complete under ≤*cd*-reductions in all other cases.*

**Proof (Sketch)** NP-completeness for ([i](#_bookmark22)) and the first three cases from ([ii](#_bookmark23)) follows from [[2](#_bookmark26)]—these are LTL-formulae.

The remaining two restricted sets in ([ii](#_bookmark23)) can be proven by a similar reduction and algorithm as for the CTL-cases.

For ([iii](#_bookmark24)), we modify the hardness part of Vardi’s proof showing that CTL٨-SAT restricted to *{*A*,* X*,* U*}* and BF is 2EXPTIME-complete [[20](#_bookmark45)]. Vardi gives a reduc- tion from the word problem for exponential-space alternating Turing machines to CTL٨-SAT(*{*A*,* X*,* U*},* BF). We restate the formulae in this reduction using either the temporal operators A*,* X and F, or the temporal operators A and U only.

* 1. *Restricting the Boolean connectives*

**Theorem 4.3** *Let T denote a set of temporal operators and let B be a set of Boolean* *functions such that* [*B*] *∈/ {*L*,* L0*}. Then* CTL٨*-*SAT(*T, B*) *is*

1. *equivalent to* CTL٨*-*SAT(*T,* BF) *if* S1 *⊆* [*B*]*,*
2. NC1*-complete under ≤*cd*-reductions if* S11 *⊆* [*B*] *⊆* M*, and*
3. TC0*-complete under ≤*cd*-reductions in all other cases.*

**Proof.** The results of Section [3](#_bookmark7) are easily generalized to CTL٨-SAT.

# Conclusion

The complexity of the satisfiability problem for temporal-operator-restricted frag- ments of CTL and CTL٨ is a trichotomy: for CTL we classified completeness for EXPTIME, PSPACE and NP, and for CTL٨ we classified completeness for 2EXPTIME, PSPACE and NP. This situation is depicted as a lattice in Figure [1](#_bookmark3).

Concerning the restrictions on the set of Boolean functions we observe a tetra- chotomy: a line from BF down to S1, whose complexity is determined by the temporal operators we allow, a similar line of NC1-complete clones from M down to S11, the two clones L and L0, whose complexity is bounded above by the complexity for the clone BF, and the remaining clones—all of which are TC0-complete. The complete lattice is shown in Figure [2](#_bookmark4).

Hence, the complexity of the satisfiability problems increases along the same edges as it does in propositional logic. In particular, if *x /→ y* can be implemented then satisfiability is as difficult as if all Boolean connectives were available, whereas

else the complexity of CTL-SAT(*T, B*) drops to a very low class inside NC1 and is particularly independent of the temporal operators (except for the clones L and L0). For CTL-SAT*P* , the satisfiability problem restricted to syntactically correct for- mulae, the TC0-complete clones R1 and D are trivially satisfiable, while the clones N, V, V0, E, and E0 yield complexity results depending on the set of CTL-operators allowed: Let *B* be a set of Boolean functions such that [*B*] *∈* V*,* V0*,* E*,* E0*,* N and let *T* be a set of CTL-operators not containing AU and EU (*T ∩ {*AU*,* EU*}* = *∅*). Then CTL-SAT*P* (*T, B*) is solvable in TC0. Otherwise, if *T ∩ {*AU*,* EU*}* = *∅* then CTL-SAT*P* (*T,* V) and CTL-SAT*P* (*T,* V0) are NLOGTIME-complete; whereas CTL-SAT*P* (*T,* E) and CTL-SAT*P* (*T,* E0) are coNLOGTIME-complete. Finally, the

promise problem CTL-SAT*P* (*T,* N) is equivalent to MOD2 under *≤*dlt

proj

For CTL-SAT*P* (*T, B*), [*B*] *∈ {*V*,* V0*,* E*,* E0*,* N*}* and *T ∩ {*AU*,* EU*}* =*/*

-reductions.

*∅*, the gap

between membership in TC0 and hardness for NLOGTIME (resp. coNLOGTIME or

AC0[2]) results—intuitively speaking—from the unfortunate circumstance that the given formula is promised to be syntactically correct, but determining the influence of some literal on the satisfyability yet has to be derived from the syntactical structure. On the one hand, determining whether some guessed literal or constant is relevant to the satisfiability of some formula seems to require the capability of counting; on the other hand, it seem unlikely that some TC0-complete is reducible to CTL-SAT*P* (*T, B*). Analogous results were obtained for full branching time logic CTL٨.

Finally, the complexity of CTL-SAT(*T,* [*B*]) and CTL٨-SAT(*T,* [*B*]) for [*B*] *∈ {*L*,* L0*}* remains unclassified. Though we obtained membership in P for CTL-SAT(*T, B*) if *T* = *{*AX*}* or *T* = *{*AG*}* (the results will be included in the full version of this paper), the interplay of linearity and temporal operators eluded CTL-SAT(*T, B*) from a detailed analysis for the remaining cases. Their complexity remains an open question. Note that the result for CTL-SAT(*{*AG*}, B*) states the hitherto first subexponential upper bound for a reflexive temporal operator in connection with the *⊕*-function.

Further work should, besides closing the just mentioned complexity gap, address a detailed analysis of the model checking problem for fragments of CTL٨, as begun in [[1](#_bookmark27)] for LTL. The fragment CTL is known to generally have an efficient (polynomial time decidable) model checking problem; we consider it very interesting to determine here for which fragments space efficient or parallel algorithms exist.

# References

1. Bauland, M., M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor and H. Vollmer, *The tractability of model checking for LTL: the good, the bad, and the ugly fragments*, in: *Proceedings Methods for Modalities 5* (2007), pp. 125–140.
2. Bauland, M., T. Schneider, H. Schnoor, I. Schnoor and H. Vollmer, *The complexity of generalized satisfiability for linear temporal logic*, in: *Proceedings of the Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science **4423** (2007), pp. 48–62.
3. B¨ohler, E., N. Creignou, S. Reith and H. Vollmer, *Playing with Boolean blocks, part I: Post’s lattice with applications to complexity theory*, SIGACT News **34** (2003), pp. 38–52.
4. Clarke, E., E. A. Emerson and A. Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications*, ACM Transactions on Programming Languages and Systems **8** (1986),

pp. 244–263.

1. Cook, S. A., *The complexity of theorem proving procedures*, in: *Proceedings 3rd Symposium on Theory of Computing* (1971), pp. 151–158.
2. Emerson, E. A., “Temporal and Modal Logic,” Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics, MIT Press, Cambridge, MA, USA, 1990 pp. 995–1072.
3. Emerson, E. A. and J. Y. Halpern, *“sometimes” and “not never” revisited: On branching versus linear time*, in: *Proceedings Symposium on Principles of Programming Languages* (1983), pp. 127–140.
4. Fischer, M. J. and R. E. Ladner, *Propositional modal logic of programs*, Journal of Computer and Systems Sciences **18** (1979), pp. 194–211.
5. Ladner, R., *The computational complexity of provability in systems of modal propositional logic*, SIAM Journal on Computing **6** (1977), pp. 467–480.
6. Laroussinie, F., *About the expressive power of CTL combinators*, Information Processing Letters **54**

(1995), pp. 343–345.

1. Lewis, H., *Satisfiability problems for propositional calculi*, Mathematical Systems Theory **13** (1979),

pp. 45–53.

1. Markey, N., *Past is for free: on the complexity of verifying linear temporal properties with past*, Acta Informatica **40** (2004), pp. 431–458.
2. Meier, A., “Complexity of Temporal Logics,” Master’s thesis, Gottfried Wilhelm Leibniz Universit¨at Hannover (2007).
3. Pippenger, N., “Theories of Computability,” Cambridge University Press, Cambridge, 1997.
4. Post, E., *The two-valued iterative systems of mathematical logic*, Annals of Mathematical Studies **5**

(1941), pp. 1–122.

1. Regan, K. and H. Vollmer, *Gap-languages and log-time complexity classes*, Theoretical Computer Science

**188** (1997), pp. 101–116.

1. Schnoor, H., *The complexity of the Boolean formula value problem*, Technical report, Theoretical Computer Science, University of Hannover (2005).
2. Sistla, A. and E. Clarke, *The complexity of propositional linear temporal logics*, Journal of the ACM **32**

(1985), pp. 733–749.

1. Vardi, M. Y. and L. Stockmeyer, *Improved upper and lower bounds for modal logics of programs: Preliminary report*, in: *STOC ’85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, Lecture Notes in Computer Science, 1985, pp. 240–251.
2. Vardi, M. Y. and L. Stockmeyer, *Lower bound in full (2EXPTIME-hardness for CTLs-SAT)*, Online, available at <http://www.cs.rice.edu/~vardi/papers/ctl_star_lower_bound.pdf> (1985).
3. Vollmer, H., “Introduction to Circuit Complexity – A Uniform Approach,” Texts in Theoretical Computer Science, Springer Verlag, Berlin Heidelberg, 1999.