

Electronic Notes in Theoretical Computer Science 239 (2009) 193–211

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

An Automata-based Approach for CTL\* With Constraints

# R´egis Gascon

*INRIA Sophia-Antipolis, CNRS, I3S, UNSA.*

*2004 Route des Lucioles, BP93, F-06902 Sophia-Antipolis Cedex, France*

Email: [regis.gascon@sophia.inria.fr](mailto:regis.gascon@sophia.inria.fr)

Abstract

We introduce a general definition for a family of branching-time logics that extend CTL∗ by allowing constraints between variables at the atomic level. These constraints allow to compare values of variables at different states of the model. We define an automata-theoretic approach to solve verification problems for such extensions. Our method is based on a finite abstraction of the infinite state space and a symbolic representation of the models that generalizes several approaches used for extensions of the linear-time logic LTL with constraints. We extend and combine several constructions involving alternating tree automata.

We a∗pply this approach to prove decidability and optimal complexity results for particular instances of

CTL extensions whenever an abstraction of the models verifying a “nice” property can be computed.

These theoretical results generalize several results on LTL with constraints where such nice abstractions are used.

*Keywords:* Temporal-logic, Model-checking, Automata approach.

# Introduction

In model-checking, temporal logics are commonly used to specify properties on symbolic representations of computing systems. The atomic formulas of these logics are usually propositional variables whose truth value depends on the states of the symbolic system. This allows to specify properties on these states but not on the data that can be used in some models s.t. clocks, counters, strings. . . To overcome this problem, several extensions of the linear-time logic LTL with constraints on the data have been considered in the literature [[19](#_bookmark28),[2](#_bookmark10),[14](#_bookmark23)]. However, less results are known about their branching-time extensions.

We consider in this paper a general definition for a family of branching-time temporal logics that extend CTL∗ by adding constraints on the data. In these extensions, the atomic formulas are refined to relations between terms of the form X *...* X*x*, representing a future value of the variable *x*. So, values of the variables at different states of an execution can be compared: for instance, the formula *A*(*x* =

1571-0661/© 2009 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.2009.05.040

XX*y*) means that for every execution the current value of *x* is equal to the value of *y* two states further. Similar extensions of LTL with constraints have already been studied where often pspace-completeness results are shown (see e.g., [[19](#_bookmark28),[2](#_bookmark10),[14](#_bookmark23)]). In [[7](#_bookmark16)], the authors have extended the automata approach of [[18](#_bookmark26)] to establish this complexity bound for satisfiability and model-checking problems. We also have adapted this technique in previous works [[9](#_bookmark17),[10](#_bookmark18)]. Thus, a natural question raised by these results would be: can we generalize this automata approach to branching- time?

We answer to this question in this paper by defining a general automata approach for satisfiability of CTL∗ extended with constraints and model-checking of this logic over a class of automata whose transitions are labeled by atomic constraints of the logic. This construction relies on a symbolic abstraction of the concrete models. Unlike [[15](#_bookmark24)], we do not introduce new classes of automata but our construction re- quires non-trivial adaptations and combinations of classical constructions. Indeed, we must deal with the capability of the atomic constraints to compare values at dif- ferent states of the model. Another difficulty is that the class of constraint automata we consider for model-checking can induce infinitely branching configuration graphs. We show how to tackle this problem by defining an abstraction of the automaton behaviour.

This construction can be used to refine several results about LTL extended with constraints (for instance [[2](#_bookmark10),[7](#_bookmark16),[3](#_bookmark12)]). Indeed, we can take advantage of the fact that the models of the logics concerned can be abstracted in such a way that the correspon- dence between concrete and symbolic models can be verified easier. We prove that satisfiability and model-checking problems are in 2exptime for the CTL∗ exten- sions of such logics. Our method directly extends the different constructions for the corresponding LTL fragments cited above, as well as the symbolic representations of the models used in these constructions. As a consequence, the optimal complex- ity results for satisfiability of linear-time formulas are preserved. As expected, the construction of the automaton for the branching-time extensions is much more com- plicated than in the linear case and the complexity bound is quite high (it is already the case for CTL∗). So, these results are mainly theoretical and we do not claim that the technique would be very efficient in practice. The theoretical significance of this work comes also from the lack of results about this kind of extensions of branching-time logic compared to their linear-time fragments. Some existing work concerns for instance the verification of constraints between counters [[4](#_bookmark13),[8](#_bookmark19)] or on stack contents [[13](#_bookmark20),[12](#_bookmark21)]. However, [[4](#_bookmark13)] restricts the use of the temporal quantifiers of the logic and [[8](#_bookmark19)] the structure of automata that can be verified. Finally, in [[13](#_bookmark20)] and [[12](#_bookmark21)] the constraints are checked by adding states in the model but this technique can only be used when the relations considered form regular languages.

# A branching-time logic extended with constraints

Concrete domains. Let VAR = {*x*0*, x*1*,...* } be a countably infinite set of vari- ables. A *concrete domain* is a pair D = ⟨*D,* R⟩ where *D* is an interpretation domain

for the variables and R is a countable set of relations on the elements of *D*. We call D*-constraint* an expression of the form R(*x*1*,... , xk*) where R ∈R is a relation of the domain whose arity is *k* and *x*1*,... , xk* ∈ VAR. A *D-valuation* is a function *v* : VAR → *D* assigning a value in *D* to every variable and the satisfaction relation is defined by *v* |= R(*x*1*,... , xn*) iff (*v*(*x*1)*,... , v*(*xn*)) ∈ *R* where *R* is the relation associated to the symbol R.

CTL∗ over concrete domains. Given a concrete domain D = ⟨*D,* R⟩, the ex- tension CTL∗(D) of the branching time logic CTL∗ over the concrete domain D is defined by

*φ* ::= Т| ¬*φ* | *φ* ∧ *φ* | *φ* ∨ *φ* | E*ψ* | A*ψ*

*ψ* ::= *φ* | ¬*ψ* | *ψ* ∧ *ψ* | R(X*i*1 *xj ,... ,* X*ik xj* ) | X*ψ* | *ψ*U*ψ*

1 *k*

where *x*1*,... , xk* ∈ VAR and R ∈ R. As usual, we distinguish in CTL∗(D) state for- mulas (*φ*) and path formulas (*ψ*) and we have A*ψ* ≡ ¬E¬*ψ*. The atomic constraints R(X*i*1 *xj ,... ,* X*ik xj* ) do not involve variables but expressions called terms. A *term*

1 *k*

is made of a variable *x* prefixed by *i* ∈ N symbols X (we shortly write X*ix*) and represents the value of the variable *x* at the *i*th next state in the model. Because of terms, atomic constraints must be in the scope of a path quantifier like the other

temporal subformulas that refer to the future. We define the next length |*φ*|X of a formula *φ* to be the maximal integer *i* s.t. a term of the form X*ix* occurs in *φ*. This value corresponds to the depth we need to explore in the model to evaluate every atomic constraint in *φ*.

The models of CTL∗(D) are infinite trees of the form *T* = ⟨*N, n*0*,* −→*,* Γ⟩ where *N* is a set of nodes, *n*0 ∈ *N* is the root, −→⊆ *N* × *N* is an edge relation and Γ: *N* −→ (VAR −→ *D*) is a map that associates a valuation to each node. Moreover, the edge relation of *T* is s.t. (i) each node except the root as exactly one incoming edge (tree shape) and (ii) every node has at least one successor. A *path* in *T* is a sequence of nodes *π* = *n*0 · *n*1 ··· s.t. *ni* −→ *ni*+1 for all 1 ≤ *i <* |*π*|. We denote the *i*th suffix of a path *π* by *πi* = *ni* · *ni*+1 ··· , and the *i*th node of *π* by *π*(*i*). By property (ii), every maximal path of a CTL∗(D) model, also called *branch*, is infinite.

We introduce additional definitions making these models easier to handle in the following. The branching degree of a graph *G* is the least upper bound of the number of outgoing edges of every node (possibly *ω*) and a *d*-graph is a graph s.t. all nodes have exactly *d* outgoing edges. We arbitrarily order the successors of every node *n* of *G* whose degree is *d* by labeling the edges with elements of {0*,... ,d* − 1}.

For every *a* ∈ {0*,... ,d* − 1}, a node *n*' is the *a*-successor of *n* iff *n* −→*a n*' and every

node has at most one *a*-successor. So, the labeling allows to describe a path by

giving the sequence of labels on the edges visited. For example, the path from the

root *n*0 following *w* = 1 · 0 · 0 ··· is *n*0 −→1

*n*1 −→0

*n*2 −→0

··· (see the bold path in

Fig. [1](#_bookmark0)). Given a path *π* in a graph, we note *wπ* the sequence made of the successive edge labels encountered when following *π*, i.e. for every 0 ≤ *i* ≤ |*π*|− 1 we have

*w*(*i*)

*π*(*i*) −→ *π*(*i* + 1).

Let *φ* be a CTL∗(D) formula. Given a model *T* , a node *n* and a branch *π*

of *T* , the state satisfaction relation ⟨*T, n*⟩ |= *φ* and the path satisfaction relation

⟨*T, π*⟩ |= *φ* are defined as following (boolean cases are standard and omitted):

⟨*T, n*⟩ |= Т for every *n* ∈ *N* ,

⟨*T, n*⟩ |=E *ψ* iff there is a branch *π* = *n*0 · *n*1 ··· s.t. *n*0 = *n* and ⟨*T, π*⟩ |= *ψ*,

⟨*T, π*⟩ |= R(X*i*1 *xj ,... ,* X*ik xj* ) iff (Γ(*π*(*i*1))(*xj* )*,... ,* Γ(*π*(*ik*))(*xj* )) ∈ *R*,

1 *k* 1 *k*

⟨*T, π*⟩ |= X*ψ* iff ⟨*T, π*1⟩ |= *ψ*,

⟨*T, π*⟩ |= *ψ*U*ψ*' iff ∃*i* ∈ N s.t. ⟨*T, πi*⟩ |= *ψ*' and for every 0 ≤ *j < i*, ⟨*T, πj* ⟩ |= *ψ*. The formula *φ* is *satisﬁable* iff there is a model *T* = ⟨*N, n*0*,* −→*,* Γ⟩ s.t. ⟨*T, n*0⟩ |= *φ*.

Example 2.1 Let IPC++ [[6](#_bookmark15)] be the language defined by the following grammar: *α* ::= *x* ≡*k y* + [*c, c*'] | *x* ≡*k* [*c, c*'] | *x* = *y* | *x* = *c* | *x < c* | *x > c* | *α* ∧ *α* | ¬*α* | ∃*x. α* where *x, y* ∈ VAR, *c, c*' ∈ Z and *k* ∈ N. We also use the abbreviations *x* ≡*k y* + *c*

and *x* ≡*k c* for *x* ≡*k y* + [*c, c*] and *x* ≡*k* [*c, c*]. The models for this language are

valuations of the form *v* : VAR → Z and the satisfaction relation |= is naturally defined for equality and inequality constraints. Concerning periodicity constraints, we have *v* |= *x* ≡*k y* + [*c*1*, c*2] iff ∃*d* ∈ Z s.t. *v*(*x*) − *v*(*y*) = *c* + *dk* for some *c*1 ≤ *c* ≤ *c*2.

We consider the concrete domain induced by the language IPC++, i.e. the concrete domain whose interpretation domain is Z and whose set of relations is induced by the language IPC++. In the following, we identify this domain with the language IPC++. So, the set of CTL∗(IPC++) atomic formulas are IPC++ constraints over te rms. For instance, the formula E ((X*x > x*) ∧ (X*x* ≡2 *x* +

1)) U (A(X*x* = 0))

is a CTL∗(IPC++) formula satisfied by the model represented

in Fig. [1](#_bookmark0) (the values of *x* are given in the nodes).



*x* =2

1

0

4

3

1

0

6

6

0

1

2

0

0

0

Figure 1. A CTL∗(IPC++) model

Constraint automata. We consider the model-checking problem for the logic CTL∗(D) over a particular class of automata called D*-constraint automata*. The transitions of such automata are labeled by constraints from D between terms rep- resenting the values of the variables at the current and the next state. Formally, a D-constraint automaton is a structure A = ⟨*Q, I, F, δ*⟩ where *Q* is a finite set of locations, *I* and *F* are respectively the sets of initial and final locations, and the transition relation *δ* is a subset of *Q* × Σ × *Q* where Σ is the set of D-constraints built on terms of the form *x* and X*x* for *x* ∈ VAR.

A state of the automaton is a pair ⟨*q, v*⟩ s.t. *q* ∈ *Q* and *v* is a valuation. We

note ⟨*q, v*⟩ −→*α* ⟨*q*'*, v*'⟩ iff there is a transition ⟨*q, α, q*'⟩ ∈ *δ* and the valuation assigning

the value *v*(*x*) to *x* and *v*'(*x*) to X*x* for every *x* ∈ VAR satisfies *α*. A path in the

automaton A is a sequence of the form ⟨*q ,v* ⟩ *α*−→0 ⟨*q ,v* ⟩ *α*−→1

··· and a run is a

0 0 1 1

labeled tree ⟨*N, n*0*,* −→*,* Γ⟩ s.t. (i) Γ(*n*0)= ⟨*q*0*, v*0⟩ where *q*0 ∈ *I* and *v*0 is an initial

valuation and (ii) if Γ(*n*)= ⟨*q, v*⟩ then for every ⟨*q*'*, v*'⟩ s.t. ⟨*q, v*⟩ −→*α* ⟨*q*'*, v*'⟩ holds in

A there is a successor of *n* labeled by ⟨*q*'*, v*'⟩. Condition (ii) allows to have infinitely

many successors when the interpretation domain of D is infinite. A run is accepting iff every infinite branch has an infinite number of nodes labeled with a final state (Bu¨chi acceptance condition). A D-constraint automaton A satisfies a CTL∗(D) formula *φ* (denoted A |= *φ*) iff there exists a run *T* of A s.t. ⟨*Tv, n*0⟩ |= *φ* where *Tv* is the restriction of *T* with the map restricted to valuations.

This definition of constraint automata subsumes several well-known models s.t. counter automata, pushdown automata or lossy channel systems, to quote few ex- amples. Indeed, one can handle all the different objects of these formalisms by considering the right interpretation domain (integers or strings in these examples) and set of relations (standard arithmetic, prefix or subword relations). Note also that one can express tests/guards s.t. *x* = 0 as well as updates like X*x* ≡*k x* +1 on the same transition using conjunctions.

Example 2.2 Using IPC++-constraint automata, we can abstract the behaviour of counter automata by performing operations modulo some integer (see [[16](#_bookmark25)]). We present in Fig. [2](#_bookmark1) the abstraction of a pay-phone controller with two counters *x* and *y* (see [[5](#_bookmark14), Ex. 1]). The increments of a counter *x* are abstracted by *x <* 232 ∧ X*x* ≡232 *x* + 1 which corresponds to the encoding of integers in standard programming languages. The formula *φ*= stands for X*x* = *x* ∧ X*y* = *y* which means that no counter is modified. Messages are omitted because they are irrelevant here.

*x <* 232 ∧ X*x* ≡232 *x* +1 ∧ X*y* = *y y* ≤ *x* ∧ X*y* ≡232 *y* +1 ∧ X*y > y* ∧ X*x* = *x*

*x*



X*x* ≡232 *x* +1 ∧ X*x > x* ∧ X*y* = *y*

*q*1

*x* =0 ∧ *y* =0

*q*2

*φ*=

*x >* 0 ∧ *φ*=

= *y* ∧ X*x* =0 ∧ X*y* =0

*q*6

*φ*=

*q*3

*φ*=

*q*5

*q*4

*y* ≤ *x* ∧ *φ*=

X*y* ≤ *x* ∧ X*y* ≡232 *y* +1 ∧ X*y > y* ∧ X*x* = *x*

Figure 2. An IPC++-constraint automaton

The *model-checking problem* takes as input a CTL∗(D) formula *φ* and a D- constraint automaton A and consists in checking whether A |= *φ*. Unlike for its LTL fragment restriction (see e.g, [[7](#_bookmark16)]), this problem cannot be reduced to the satisfiability problem because there is no way in the logic to express that a state have infinitely many successors.

# From concrete to symbolic models

We consider concrete domains of the form D = ⟨*D,* R⟩ s.t. *D* is infinite (otherwise CTL∗(D) can easily be reduced to CTL∗) and the set R is not trivial, i.e. it contains at least one relation that is neither the empty nor the universal set. Let *φ* be a

CTL∗(D) formula s.t. *V* ⊆ VAR is the finite set of variables used in *φ* and |*φ*|X = *l*. We denote the set of terms used in *φ* by Terms = {X*ix* | *x* ∈ *V* and *i* ∈ {0*,... , l*}}. Symbolic valuations. The automata-based approach we propose relies on an abstraction of the models w.r.t. *φ*. The first step is to abstract valuations of the form Terms → *D*. We call *symbolic valuation abstraction* w.r.t. *φ* a pair composed of a set SV(*φ*) and a surjective function of the form *symb* : (Terms → *D*) → SV(*φ*) associating a unique element of SV(*φ*) to every valuation *v* : Terms → *D* , which verifies the following condition:

(SV) for every pair of valuations *v, v*' s.t. *symb*(*v*) = *symb*(*v*') and every atomic constraint *α* of *φ*, we have *v* |= *α* iff *v*' |= *α*.

The above condition implies that each element of SV(*φ*), called *symbolic valuation*, is an equivalence class of valuations. Thus, we can define a symbolic satisfaction relation as following: for every atomic constraint *α* of *φ* and symbolic valuation *sv* ∈ SV(*φ*), *sv* |=symb *α* iff for every valuation *v* s.t. *symb*(*v*)= *sv* we have *v* |= *α*.

Example 3.1 There are different methods to define such symbolic valuation ab- stractions. A famous one is the region abstraction of timed automata [[1](#_bookmark11)]. Several works about extensions of LTL over concrete domains also relies on symbolic val- uation abstractions [[2](#_bookmark10),[7](#_bookmark16),[3](#_bookmark12)]. In most of them the symbolic valuations are described by sets of constraints.

For instance, given a CTL∗(IPC++) formula *φ* we denote respectively by Terms and CONS the sets of terms and constants used in *φ*, and *K* the least common multiple of the set of integers *k* s.t. a relation ≡*k* occurs in *φ*. We define SV(*φ*) to be the set of triples of the form ⟨*X*eq*, X*cons*, X*mod⟩ s.t.

* *X*eq is a maximal consistent set of equality relations between the elements of Terms. Maximal consistency means that *X*eq is consistent, i.e. there is a valuation that satisfies all the constraints, and that no proper extension of this set is consis- tent.
* *X*cons is a maximal consistent set of comparisons between the elements of Terms and CONS. For every *t* ∈ Terms and *c* ∈ CONS there is a constraint *t* ∼ *c* ∈ *X*cons where ∼∈ {*<,* =*, >*}.
* *X*mod is a maximal consistent set of periodicity relations of the form *t* ≡*K c* s.t. *t* ∈ Terms and *c* ∈ {0*,... K*}. So, for every *t* ∈ Terms there is exactly one constraint of the form *t* ≡*K c* in *X*mod.

Simple additional properties that we do not develop here (transitivity, reflexivity, consistence of the modulos and comparisons. . . ) have to be checked in order to ensure the consistency of *X*eq *X*cons *X*mod. The map *symb* associates to every valuation *v* : Terms → Z the element of SV(*φ*) s.t. *v* satisfies every constraint in the triple. This element is unique since the definition of SV(*φ*) induces a partition of ZTerms. Then, the property (SV) can be shown by structural induction on the constraint in the same way that [[6](#_bookmark15), Lemma 1].

For the sake of simplicity, we will consider in the following that symbolic val- uations are sets of relations between terms. The definitions in the remaining can be rewritten with more general notations but it would make them less clear. More-

over, all the results that we want to generalize in Sect. [4](#_bookmark6) use such a representation of symbolic valuations.

Frames. Symbolic valuations only give information on a linear path whereas the models of CTL∗(D) are trees. So we have to extend this symbolic representation to a branching structure giving information on the different paths. In order to consider only a finite set of such branching structures, we first establish a bound property on the branching degree of the models that have to be considered.

A CTL∗(D) formula is in *positive normal form* iff negations appear only in atomic constraints. The set of positive formulas can be defined by

*φ* ::= Т| E*ψ* | A*ψ* | *φ* ∧ *φ* | *φ* ∨ *φ*

*ψ* ::= *φ* | R(X*i*1 *x*1*, ...,* X*ik xk*) | ¬R(X*i*1 *x*1*, ...,* X*ik xk*) | *ψ* ∧ *ψ* | *ψ* ∨ *ψ* | X*ψ* | *ψ*U*ψ* | *ψ*U˜*ψ*

where U˜ is the dual operator of U, i.e. *φ*U˜*φ*' ≡ ¬(¬*φ*U¬*φ*'). One can easily build

from any CTL∗(D) formula an equivalent positive formula by using the duality of

CTL∗(D) logical operators and quantifiers.

Lemma 3.2 *For every* CTL∗(D) *formula φ there is a formula φ*' *in positive normal form computable in linear time s.t. φ is satisﬁable iff φ*' *is satisﬁable.*

For every formula *φ*, we denote by E(*φ*) the number of existential path quan- tifiers in the positive normal form of *φ*. The following Lemma is the same kind of small model property shown in [[11](#_bookmark22)] for CTL∗.

Lemma 3.3 *A* CTL∗(D) *formula φ is satisﬁable iff there exists a* (E(*φ*)+ 1)*-tree that satisﬁes φ.*

Basically, given a model satisfying *φ* we can build another model enjoying this property by unfolding at each step one path satisfying each existential subformula of the positive normal form of *φ* which is satisfied in the initial model. By definition, there are at most (E(*φ*) + 1) such subformulas. Then we complete to reach (E(*φ*)+

1) outgoing paths for every node (see details in Appendix [A](#_bookmark29)).

Let us set *d* = E(*φ*). The result of Lemma [3.3](#_bookmark5) is very important since it allows us to restrict the search of a model satisfying *φ* to the subclass of (*d* + 1)-trees. Given a symbolic valuation abstraction ⟨SV(*φ*)*, symb* ⟩, we define a *frame fr* as a mapping {0*,... , d*}*l* → SV(*φ*) associating to every word of length *l* (corresponding to paths) a symbolic valuation. As we said before, we suppose that the elements of SV(*φ*) are sets of constraints (see Ex. [3.1](#_bookmark3)). The mapping *fr* must coincide on the common prefixes, i.e. constraints that refer to the same terms in different symbolic valuations must be the identical:

(FR) for every *w*1*, w*2 ∈ {1*,... , d*}*l* and *w* their longest common prefix, we have *α* ∈

*fr* (*w*1) iff *α* ∈ *fr* (*w*2) for every atomic constraint *α* of the form R(X*i*1 *x*1*,... ,* X*ik xk*)

s.t. 0 ≤ *i*1*,... , ik* ≤ |*w*|.

We denote by Frame(*φ*) the set of frames defined w.r.t. *φ*.

Symbolic models. We use frames to define a symbolic representation of CTL∗(D) models. We say that a pair of frames ⟨*fr, fr* '⟩ is *a-step consistent* iff for every

*i*1*,... , ik >* 0, we have R(X*i*1 *x*1*,... ,* X*ik xk*) ∈ *fr* (*a* · *w*) ⇔ R(X*i*1−1*x*1*,... ,* X*ik* −1*xk*) ∈ *fr* '(*w* · *b*) for every *b* ∈ {1*,... , d*}. This means that all the symbolic valuations associated with path moving in the direction of *a* are consistent with all the possible next ones (similarly to (FR)).

A *symbolic model T*symb = ⟨*N,* {*n*0}*,* −→*,* Γsymb⟩ is a labeled *d*-tree associat- ing to each node a frame (Γsymb : *N* → Frame(*φ*)) s.t. every node *n* of *T*symb is *one-step consistent* : for every *a* ∈ {1*,... , d*} the *a*-successor *n*' of *n* is s.t.

⟨Γsymb(*n*)*,* Γsymb(*n*')⟩ is *a*-step consistent. The symbolic satisfaction relation defined for symbolic valuations can be naturally extended using the satisfaction relation of CTL∗(D). The only difference is for the atomic constraints: for any branch *π* of *T*symb we have ⟨*T*symb*, π*⟩ |=symb *α* iff Γsymb(*n*)(*wπ*) |=symb *α* where *n* is the initial node of *π*.

We also define a satisfaction relation between concrete models and symbolic models. Given a graph *G* = ⟨*N,* {*n*0}*,* −→*,* Γ⟩, a node *n* ∈ *N* and a path *π* ∈ *Nl*, *vπ* is the valuation s.t. for every *x* ∈ VAR and *i* ∈ {0*,... , l*}, *vπ*(X*ix*)= Γ(*π*(*i*))(*x*). We note ⟨*G, n*⟩ |= *fr* iff for every *w* ∈ {1*,... , d*}*l* the path *π* starting at *n* and following *w* is defined in *G* and *symb*(*vπ*)= *fr* (*wπ*). A graph *G* satisfies a symbolic model *T*symb iff for every node *n*symb reachable in *T*symb from the initial state by following a word *w* ∈ {1*,... , d*}∗ the node *n* reachable from the root by following the same path in *G* is s.t. ⟨*G, n*⟩ |= Γsymb(*n*symb). This relation express that *T*symb is an abstraction of *G* but it also implies a correspondence between the nodes of *G* and *T*symb. Indeed, *G* must be a *d*-graph (otherwise some required path can be undefined) and when following the same path in both graph the nodes reached verify the satisfaction relation. The result below, which is the base of the automaton construction, can be proved by unfolding the definitions of this section.

Lemma 3.4 *A* CTL∗(D) *formula φ is satisﬁable iff there exist a symbolic model*

*T*symb *and a d-tree T s.t. T*symb |=symb *φ and T satisﬁes T*symb*.*

# A decidable model-checking problem

Nice abstractions. Several works have studied extensions of LTL with constraints from a concrete domain D, which is the linear-time fragment of CTL∗(D), where decidability proofs often rely on the kind of abstraction described in Sect. [3](#_bookmark2). We have observed from these works that, in many cases, checking the existence of a concrete model satisfying a given symbolic model can be done by performing tests only on a finite number of consecutive symbolic valuations. Often, checking one-step consistence is enough. This is the case for instance for extensions with concrete do- mains of the form ⟨*D, <,* =⟩ which are dense and open (see [[7](#_bookmark16)]) or domains verifying the global consistency of [[2](#_bookmark10)] or the *ω*-admissibility of [[17](#_bookmark27)]. In artificial intelligence, the domain RCC8 allowing to specify topological relations on the real plane R2 or the Allen relations on rational numbers used to represent temporal interval knowl- edge are some examples of such concrete domains among many others (see a more exhaustive list in [[2](#_bookmark10), Sect.2]). These domains can also be used in temporal logic extensions to express interesting properties about space and time representations

of the knowledge. According to our motivations, we are interested in extending the results about extensions of LTL with the domains ⟨Q*, <,* =⟩ and ⟨R*, <,* =⟩ [[7](#_bookmark16)], IPC++ [[6](#_bookmark15)] (see Ex. [4.2](#_bookmark8)) or the qualitative spatial reasoning of [[2](#_bookmark10)]. Finally, the re- sults for the LTL extension of [[3](#_bookmark12)] with constraints built from the separation logic to verify memory allocation of programs also use an abstraction technique verifying this property. For all these LTL extensions, the satisfiability and model-checking problems have been shown to be pspace-complete. However, there are no corre- sponding results concerning their branching-time extension. Herein, we refine these results by extending the automata based approach introduced in [[7](#_bookmark16)].

First, we want to introduce a general property to subsume all the abstractions used in the above mentioned works. A symbolic valuation abstraction ⟨SV(*φ*)*, symb* ⟩ is said “*nice*” iff for every symbolic valuation *sv* ∈ SV(*φ*) any partial assignment of the terms *v*' which satisfies all the constraints of *sv* involving only terms in the definition domains of *v*' can be extended to a valuation satisfying all the constraints of *sv* . In other words, any partial assignment that does not contradict any constraint can be extended to a valuation satisfying all the constraints. This property directly implies the following result and simplifies the forthcoming developments.

Lemma 4.1 *For every nice symbolic abstraction, every one-step consistent sym- bolic model is satisﬁable.*

Nice symbolic abstractions also satisfy the property that any frame is satisfiable. This is not the case in general even if the symbolic valuations associated to each path are satisfiable.

Example 4.2 The abstraction we have defined for CTL∗(IPC++) models can easily be proved to be nice. Informally, suppose that we are given a symbolic valuation and we know a partial valuation of the terms satisfying the hypothesis. Let *t* be a term we want to assign a value. If there is a constraint *t* = *t*' in *X*eq s.t. *t*' has already a value assigned or *t* = *c* in *X*cons then we are done. Otherwise one can assign a value satisfying the constraint *t* ≡*K c* in *X*mod in the interval defined by the constraints of *X*cons since symbolic valuations are supposed to be consistent. By repeating this procedure, we obtain a valuation satisfying the whole symbolic valuation.

Consider a formula *φ* of CTL∗(D) with a nice symbolic abstraction ⟨SV(*φ*)*, symb* ⟩ and a D-automaton A = ⟨*Q, I, F, δ*⟩. Wlog, we suppose that *φ* is in positive form (thanks to Lemma [3.2](#_bookmark4)). We note *V* ⊆ VAR the variables used in *φ* and A, |*φ*|X = *l* and E(*φ*)= *d*.

Satisfiability. First, let us say few words about checking whether *φ* is satisfiable. By Lemma [3.4](#_bookmark7), we can solve this problem by defining an automaton A*φ* accepting

*d*-trees which is the intersection of two automata: A*φ*

symb

checking the symbolic

satisfaction of the formula and A*φ*

sat

checking whether the input corresponds to a

satisfiable symbolic model. In a nutshell, the automaton A*φ*

symb

can be defined by

adapting the automaton construction for CTL∗ using the same ideas as in [[15](#_bookmark24)]. The

main technical difficulty is to take into account the fact that atomic constraints refer to future values and paths from the current state to these values may need to be stored. This can be done easily by storing informations in the control states of the automata because the number of possible paths is finite (details in Appendix [D](#_bookmark30)).

Since we have a nice abstraction, A*φ*

sat

just have to check one-step consistency.

This automaton can be built by taking as transition relation the elements s.t. the next frame is consistent with the current one w.r.t. the move in the tree. So the construction requires an exponential number of tests.

By construction, the language accepted by A*φ* is non-empty iff *φ* is satisfiable (consequence of Lemma [3.4](#_bookmark7)). Since the emptiness problem for alternating tree automata is decidable (exptime-complete), we have all the elements to establish decidability. Moreover, a simple complexity analysis allows to prove that the con-

struction of A*φ*

symb

can be done in exponential time whenever the symbolic satis-

faction of an atomic constraint can be checked in exponential time. This is the case of the abstractions used for every linear-time logic cited at the beginning of this section (the symbolic satisfaction relation in CTL∗(IPC++) can be tested in ptime). So the complexity of the whole procedure for these logics is 2exptime. The 2exptime-hardness is easy to get since CTL∗ is subsumed by CTL∗(D) when R is not trivial (propositional variables can be simulated).

Theorem 4.3 CTL∗(D) *satisﬁability problem is* 2exptime*-complete if for every formula there is a nice abstraction s.t. symbolic satisfaction relation can be tested in* exptime*.*

Model-checking. We now consider the model-checking problem for CTL∗(D). The main difficulty we have to handle is that D-constraint automata can induce infinitely branching configuration graphs. So we start by defining an abstraction that bounds the branching degree and then we will extend the approach used for satisfiability. The difference with the method we sketched for satisfiability is that we have to recognize only the set of symbolic models that are satisfied by a run of A.

We extend slightly the definition of the abstraction we consider. We note SV(*φ,* A) the set of symbolic valuations that now have to be built w.r.t. the atomic constraints of the formula *φ* and the constraints on the transitions of the automa- ton A. This abstraction still have to verify the condition (SV) so that we can also define an associated symbolic satisfaction relation as in Sect. [3](#_bookmark2).

Lemma 4.4 *For every* CTL∗(D) *formula φ and* D*-automaton* A*, one can build an*

*automaton* A*r such that* A |= *φ iff there is a symbolic model T*symb *accepted by*

abs

*r*

A

abs

*and verifying T*symb |=symb *φ.*

We explain in few words how this automaton can be built. The whole construc- tion is given in Appendix [E](#_bookmark31). We introduce the projection of SV(*φ,* A) on the set of variables denoted by SV0(*φ,* A). This set is obtained by keeping the relations refer- ring to variables of the current state only which means that *X* ∈ SV0(*φ,* A) iff there exists *sv* ∈ SV(*φ,* A) s.t. for every *x*1*,... , xn* ∈ VAR we have R(*x*1*,... , xn*) ∈ *X* iff

R(*x*1*,... , xn*) ∈ *sv* . The elements of SV0(*φ,* A) are then used to abstract the states of A: the state ⟨*q, v*⟩ of A corresponds to ⟨*q, X*⟩ in an intermediate automaton Aabs

s.t. *X* ∈ SV0(*φ,* A) and *v* satisfies all the constraints in *X*. We build the automa- ton Aabs in such a way that each path in A corresponds to a one-step consistent sequence in Aabs. Moreover, Aabs generates finite branching runs only. From this

automaton we can then define the automaton A*r*

abs

recognizing the runs generated

by Aabs and establish the result of Lemma [4.4](#_bookmark9).

So we modify the construction for the satisfiability problem by intersecting the

automaton A*φ* with the alternating tree automaton A*r*

abs

recognizing the symbolic

models corresponding to Aabs. We can easily deduce from the result of Lemma [3.4](#_bookmark7)

and the construction of A*r* that A |= *φ* iff the language accepted by A*φ* ∩ A*r* is

abs

abs

non-empty. Thus we obtain a decidability procedure.

Theorem 4.5 CTL∗(D) *model-checking problem is decidable whenever there exists a nice symbolic valuation abstraction.*

The complexity of this extended construction does not increase if the symbolic satisfaction relation can be checked in exptime.

Theorem 4.6 CTL∗(D) *model-checking problem is in* 2exptime *if for every for- mula there is a nice abstraction s.t. symbolic satisfaction relation can be tested in* exptime*.*

Concluding remarks. The gap with the complexity of CTL∗ comes from the difference of expressiveness and conciseness between D-automata and Kripke struc- tures. Though being difficult to handle in practise, these theoretical results gener- alize several works on LTL extensions and improve the knowledge about temporal logics extended with concrete domains. However, it remains other concrete domains remains like ⟨Z*, <,* =⟩ for which one need to check conditions on the whole symbolic model of CTL∗ extention. We hope that this general approach approach could be extended to such cases.

# References

1. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.
2. P. Balbiani and J. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS’02*, volume 2309 of *LNAI*, pages 162–173. Springer, 2002.
3. R. Brochenin, S. Demri, and E´. Lozes. Reasoning about sequences of memory states. In *Proceedings of LFCS’07*, volume 4514 of *LNCS*, pages 100–114. Springer, 2007.
4. K. Cˇerans. Deciding properties of integral relational automata. In *ICALP*, volume 820 of *LNCS*, pages 35–46. Springer, 1994.
5. H. Comon and V. Cortier. Flatness is not a weakness. In *CSL*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2000.
6. S. Demri. LTL over integer periodicity constraints. In *Proceedings of FoSSaCS’04*, volume 2987 of

*LNCS*, pages 121–135. Springer, 2004.

1. S. Demri and D. D’Souza. An automata-theoretic approach to constraint LTL. In *FST&TCS’02*, volume 2256 of *LNCS*, pages 121–132. Springer, 2002.
2. S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a model-checker for counter systems. In *ATVA’06*, volume 4218 of *LNCS*, pages 493–507. Springer, 2006.
3. S. Demri and R. Gascon. Verification of qualitative Z-constraints. In *CONCUR’05*, volume 3653 of

*LNCS*, pages 518–532. Springer, 2005.

1. S. Demri and R. Gascon. The effects of bounding syntactic resources on Presburger LTL (extended abstract). In *TIME’07*, pages 94–104. IEEE Computer Society Press, 2007.
2. E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *I&C*, 61(3):175–201, 1984.
3. J. Esparza, A. Kuˇcera, and S. Schwoon. Model checking LTL with regular valuations for pushdown systems. *I&C*, 186(2):355–376, 2003.
4. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems (extended abstract). In F. Moller, editor, *INFINITY’97*, volume 9 of *ENTCS*. Elsevier Science Publishers, 1997.
5. D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. On the computational complexity of spatio-temporal logics. In *FLAIRS’03*, pages 460–464, 2003.
6. O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *JACM*, 47(2):312–360, 2000.
7. G. Logothetis and K. Schneider. Symbolic model checking of real-time systems. In *TIME*, pages 214–223, 2001.
8. C. Lutz and M. Milicic. A tableau algorithm for description logics with concrete domains and gcis. In

*TABLEAUX 2005*, LNAI, 2005.

1. M. Vardi and P. Wolper. Reasoning about infinite computations. *I&C*, 115:1–37, 1994.
2. F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In

*KR’00*, pages 3–14, 2000.

# Proof of Lemma [3.2](#_bookmark4)

Let *φ* be a CTL∗(D) formula. We define a normalization of the formula that push the negations at the atomic level. The map *f* is defined by induction on the structure

* + *f* (¬¬*φ*)= *f* (*φ*)*,*
  + *f* (¬(*φ* ∧ *φ*')) = *f* (¬*φ*) ∨ *f* (¬*φ*')*,* • *f* (¬(*φ* ∨ *φ*')) = *f* (¬*φ*) ∧ *f* (¬*φ*')*,*
  + *f* (¬E*φ*)= A*f* (¬*φ*)*,* • *f* (¬A*φ*)= E*f* (¬*φ*)*,*

of *φ*.

* *f* (¬X*φ*)= X*f* (¬*φ*)*,*
* *f* (¬(*φ*U*φ*')) = *f* (¬*φ*)U˜*f* (¬*φ*')*,* • *f* (¬(*φ*U˜*φ*')) = *f* (¬*φ*)U*f* (¬*φ*')*,*
* for the remaining cases, f coincides with the identity.

For every *φ*, the computation of *f* (*φ*) terminates since the recursive calls are on strict subformulas. By construction, the only subformulas that can be negated are the atomic formulas. So, *f* (*φ*) is in positive form. Finally, it is easy to prove that *φ* is satisfiable iff *f* (*φ*) is also satisfiable since the definition of *f* respects logical equivalences.

# Proof of Lemma [3.3](#_bookmark5)

Let *φ* be a CTL∗(D) formula and *d* = E(*φ*) + 1. Wlog, we can suppose that *φ* is already in positive form since it does not change its set of models. We denote by

{E*ψ*1*,... ,* E*ψd*} the set of existential subformulas in *φ*.

Suppose that a graph *G* = ⟨*N,* {*n*0}*,* −→*,* Γ⟩ satisfies *φ*. It is well known that this model can be unfolded into a tree-like model. We adapt this unfolding so that each existential formula satisfied at a node of *G* is satisfied along a designated path in our construction.

The tree *T* = ⟨*N* '*,* {*n*' }*,* −→'*,* Γ'⟩ where *N* ' ⊆ *N* × N and *n*' = ⟨*n*0*,* 0⟩ is defined

0 0

as following. We define the map Γ' such that Γ'(⟨*n, i*⟩)= Γ(*n*) for every ⟨*n, i*⟩ ∈ *N* '. We describe the construction of *T* by induction on the distance from the root of *T* .

Now, suppose that we have already built a path from *n*' to *n*' = ⟨*n, i*⟩ in *T*

0

and let {E*ψn,... ,* E*ψn*} be the set of existential formulas, satisfied by the node *n*

1 *k*

in *G*. For every formula E*ψj* of this set, there exists a path *π* = *nj* · *nj* · *nj* ··· in

0 1 2

*G* such that *nj*

0

= *n* satisfying E*ψj*. We add a copy of this path *T* by adding the

*j j j* 0 *j*

edges ⟨*n, i*⟩ −→ ⟨*n*1*,i* + 1⟩ and ⟨*nl ,i* + *l*⟩ −→ ⟨*nl*+1*,i* + *l* + 1⟩ for every *l >* 0. This

method ensure that two paths that we copy never overlap. Obviously, the number

of paths added by this operation is bounded by *d*. We can then add additional copies of paths so that ⟨*n, i*⟩ has exactly *d* successors (this does not change the set of existential formulas satisfied).

It is easy to prove that ⟨*T, n*' ⟩ |= *φ* by induction on the structure of the formula

0

.

# Proof of Lemma [3.4](#_bookmark7)

Let *φ* be a satisfiable CTL∗(D) formula. By Lemma [3.3](#_bookmark5) there is a *d*-tree *T* =

⟨*N,* {*n*0}*,* −→*,* Γ⟩ such that ⟨*T, n*0⟩ |= *φ* (where *n*0 is the root of *T* ). We can define inductively a symbolic model *T*symb = ⟨*N* '*,* {*n*' }*,* −→*,* Γsymb⟩ satisfied by *T* . We use a bijection between the nodes of *T* and *T* ': we denote by {*n*0*, n*1*,...* } the set of nodes of *T* and we prime the corresponding nodes of *T*symb. For every *n* ∈ *N* and sequence *w*, we define *vn,w* the valuation s.t. for every *x* ∈ VAR we have

0

*v* (X*ix*) = Γ(*π*(*i*))(*x*) where *π* is defined by *π*(0) = *n* and *π*(*i*) *w*−→(*i*)

*n,w*

every *i* ∈ N.

*π*(*i* + 1) for

* + The root of *T*symb is the node *n*' and Γsymb is such that Γsymb(*n*' )(*w*)= *symb*(*vn ,w*)

0 0 0

for every *w* ∈ {1*,... , d*}*l*.

* + For each node *n*' of *T*symb we define the next level as following. For every *a* ∈

{1*,... , d*}, we add an *a*-successor *n*' of *n*' in *T*symb. The map Γsymb is extended

*a*

on the new node by defining Γ(*n*' )(*w*) = *symb*(*vn ,w*) for every *w* ∈ {1*,... , d*}*l*,

*a a*

where *na* is the *a*-successor of *n* in *T* .

The one-step consistency of *T*symb and the correctness of the frames w.r.t. property (FR) can be deduced from the fact that the successive symbolic valuations are built from paths that overlaps and shares common valuations. In this construction we clearly have a one-one correspondence between the nodes of *T* and the nodes of *T*symb. We also have *T* |= *T*symb by definition of the successive frames (for every node *n* of *T* , we have ⟨*T, n*⟩ |= Γsymb(*n*')).

Let *α* be an atomic constraint occurring in *φ*. Using property (SV) of symbolic valuations, for every node *ni* of *T* , *w* ∈ {1*,... , d*}*l* and for any valuation *v*' satisfying

*symb*(*vn ,w*) = *symb*(*v*') we have *vn ,w* |= *α* iff *v*' |= *α*. This implies that for every

*i* *i*

*ni* in *T* and *w* ∈ {1*,... , d*}*l*, if *vn ,w* |= *α* then Γsymb(*n*')(*w*) |=symb *α*. Thus, for

*i* *i*

every path *π* = *ni*0

· *n*

*i*1

· *ni*1

··· in *T* , ⟨*T, π*⟩ |= *α* implies ⟨*T*symb*, π*'⟩ |=symb *α* where

*π*' = *n*'

*i*

0

' ··· .

Then it is easy to prove that *T*symb |=symb *φ* since the symbolic satisfaction rela-

tion differs from CTL∗(D) satisfaction relation only for atomic constraints and we have a one-one correspondence between the nodes of *T* and those *T*symb.

Conversely suppose that *T*symb |=symb *φ* and *T* |= *T*symb. We recall that by definition of *T* |= *T*symb there must exist a bijection between the nodes of *T*symb and *T* .

We consider a constraint *α* of *φ*, a branch *π* of *T*symb and we pose *w* to be the prefix of length *l* of *wπ*. If we have ⟨*T*symb*, π*⟩ |=symb *α* for an atomic constraint *α* then by definition of the symbolic valuation Γsymb(*π*(0))(*w*) satisfies *α*. Since *T* |= *T*symb then the node *n* corresponding to *π*(0) in *T* is such that ⟨*T, n*⟩ |= Γsymb(*π*(0)) which implies that *vn,w* |= Γsymb(*π*(0))(*w*). By definition of the symbolic satisfaction relation we have *vn,w* |= *α*. Thus we have a corresponding branch *π*' in *T* such that

⟨*T, π*'⟩ |= *α* where *π*' is the path starting at the node *n* and following the same directions than *π*.

Since the symbolic satisfaction relation only differs from CTL∗(D) satisfaction relation at the atomic level we can then prove that ⟨*T, n*0⟩ |= *φ* using the correspon- dence between the nodes of *T* and *T*symb.

# Construction of A*φ*

symb

The automaton A*φ*

symb

is an alternating parity tree automaton. We use the follow-

ing notations for alternating tree automata: an alternating *d*-tree automaton A is denoted by a tuple ⟨*Q, Q*0*,* Σ*, δ,* Cond⟩ where *Q* is a set of states, Σ is the input alphabet, *Q*0 ⊆ *Q* is a set of initial states, Cond specifies the acceptance condition and the transition relation *δ* is a subset of ⊆ *Q* × Σ × PBF({1*,... , d*}× *Q*). The set PBF(*X*) of positive boolean formula is defined by:

*θ* ::= Т|⊥ | *p* | *θ* ∧ *θ* | *θ* ∨ *θ*

where *p* is an element of *X*. We say that a set *Y* ⊆ *X* satisfies a formula *θ* ∈ PBF(*X*) iff the valuation assigning true to the elements of *Y* and false to the other elements of *X* saisfies *θ*. The transition relation associates to a pair ⟨*q, α*⟩ ∈ *Q* × Σ a posi- tive boolean formula over {1*,... , d*}× *Q* which stands for the direction we move in the tree and the state we move in the automaton. We note *δ*(*q, α*) = *θ* whenever

⟨*q, α, θ*⟩ ∈ *δ*. For instance, *δ*(*q, α*)= ⟨0*, q*1⟩∨ (⟨1*, q*2⟩∧ ⟨1*, q*3⟩) means that if we are in the state *q* and we read the letter *α* then either we move in the 0-succesor of the current node of the tree and the automaton moves in the state *q*1 or we send

two a copy of the automaton moving in the 0-succesor of the current node, the first one entering in state *q*2 and the other in state *q*3. A run of A on a *d*-tree *T* = ⟨*N,* {*n*0}*, δ,* Γ⟩ is another tree *Tr* = ⟨*Nr,* {*n*0*,r*}*,* −→*r,* Γ*r*⟩ such that the root is labeled by a state *q*0 ∈ *Q*0 and the other nodes by an element of *N* ×*Q*. A node of *Tr* labeled by ⟨*q, n*⟩ corresponds to a copy of the automaton in state *q* and reading the node *n*. Each node of *Tr* and its successors have to satisfy the transition property: for every node *nr* of *Tr* such that Γ*r*(*nr*)= ⟨*q, n*⟩ and *δ*(*q,* Γ(*n*)) = *θ*, the (possibly empty) set *S* ⊆ ({1*,... , d*}× *Q*) defined by ⟨*a, q*'⟩∈ *S* iff there is a successor labeled by ⟨*n*'*, q*'⟩ ∈ *S* where *n*' is the *a*-successor of *n* in *T* satisfies *θ*.

We describe the construction of A*φ*

symb

by induction on the structure of the

formula *φ*. The basis case for *φ* ≡Т is obvious.

We say that a CTL∗(D) formula *φ*' is maximal in *φ* iff *φ*' is a strict subformula of *φ* and there is no strict subformula *φ*'' of *φ* such that *φ*' is a strict subformula of *φ*''. We denote by max(*φ*) = {*φ*1*,... , φn*} the set of strict subformula of *φ*. With each maximal subformula *φi* of *φ* we suppose by induction hypothesis that the

automaton A*φi* = ⟨*Qi,* {*qi* }*,* Frame(*φ*)*, δi,* Rank*i*⟩ and its complement, denoted

symb

0

by A˜*φi*

˜*i* *i*

˜*i* ˜ *i*

symb = ⟨*Q ,* {*q*˜0}*,* Frame(*φ*)*, δ ,* Rank ⟩, are defined. We assume wlog that the

states of these automata are disjoint.

If *φ* ≡ *φ*1 ∧ *φ*2 then A*φ*

symb

= ⟨*Q*1 ∪ *Q*2 ∪ {*q*0}*,* {*q*0}*,* Frame(*φ*)*, δ,* Rank⟩ is defined

as following. Intuitively we add a new initial state and send copies to A*φ*1

symb

and

*φ*2

A

symb

. So the functions *δ* and Rank agrees with *δ*1 and Rank1 (resp. *δ*2 and

Rank2) on the states of *Q*1 (resp. *Q*2). For the state *q*0 we pose Rank(*q*0) = 1

and *δ*(*q*0*, fr* )= *δ*(*q*1*, fr* ) ∧ *δ*(*q*1*, fr* ). The case *φ* ≡ *φ*1 ∨ *φ*2 is similar with *δ*(*q*0*, fr* )=

0 0

*δ*(*q*1*, fr* ) ∨ *δ*(*q*1*, fr* ).

0 0

If *φ* ≡ *Eψ* we adapt the linear construction for *ψ* by verifying a single branch of

the tree. The adaptation is not straightforward because at every position we have to store the *l*th next step to evaluate the atomic constraints. We describe below the whole construction. We define *cl* (*φ*) the closure of *φ* as usual with the only difference that maximal subformulas are considered as propositional variables. An atom of *φ* is a maximally consistent subset of *cl* (*φ*) and the set of atoms of *φ* is denoted by Atom(*φ*). Let A1 = ⟨*Q,* {*q*0}*,* Frame(*φ*) × 2max(*φ*)*, δ,F* ⟩ be the generalized Bu¨chi alternating tree automaton s.t.

* *Q* = ⟨Atom(*φ*)*,* {0*,... , d*}*l*⟩∪ {*q*0},
* *δ*(*q*0*,* ⟨*fr,X*⟩)= *π*∈*σl δ*(⟨At*, π*⟩*,* ⟨*fr ,* X*,* )⟩ s.t. *φ* ∈ At,
* *δ*(⟨At*,a* · *π*⟩*,* ⟨*fr,X*⟩)= *b*∈{0*,...,d*}(*a,* ⟨At'*,π* · *b*⟩) iff
  + *fr* (*a* · *π*) |= *α* for every *α* ∈ At,
  + *φi* ∈ *X* for every *φi* ∈ At,
  + X*φ* ∈ At iff *φ* ∈ At'.
* let {*φ*' U*φ*''*,... , φ*' U*φ*''} be the set of until formulae in *cl* (*φ*). We pose *F* =

1 1 *r r*

{*F*1*,... , Fr*} where for every *i* ∈ {1*,... , r*}, *Fi* = {*At* ∈ *Q* : *φ*'U*φ*'' /∈ At or *φ*'' ∈

At}.

*i i* *i*

The idea is that at every position we know the *l* next moves to make in the tree and when we fire a transition in the automaton we follow the path that has been already guessed and guess the (*l* + 1)thth next step.

We can easily transform A1 into an equivalent automaton A2 = ⟨*Q*'*,* {*q*' }*,* Frame(*φ*)× 2max(*φ*)*, δ*'*,* Rank'⟩ with a parity acceptance condition instead of the generalized

0

Bu¨chi condition. To build A*φ*

symb

we need to complete this construction in order

to handle the maximal subformulas supposed to be true at each step. This can be done by applying the following rules to the transition relation of A2

0

*δ*(*q, fr*)=

⟨*fr,X*⟩∈Frame(*φ*)×2max(*φ*)

*δ*'(*q,* ⟨*fr,X*⟩) ∧

*φi*∈*X*

*δi*(*qi , fr* ) ∧

*φi*/∈*X*

0

*δ*˜*i*(*q*˜*i , fr* )

For the case *φ* = *Aψ* we build the automaton for *E*¬*φ* and then complement it.

Now we prove the correctness of this construction. We develop the only non trivial case which is *φ* ≡ *Eψ*. If a symbolic model *T*symb satisfies *φ* then there is a

path *π* such that ⟨*T*symb*, π*⟩ |=symb *ψ*. By construction a run of A*φ*

symb

that proceeds

along *π* must satisfy *φ*. Conversely, if a symbolic model *T*symb is accepted by a run

*Tr* of A*φ*

symb

then there is a path *π* in *Tr* such that if we proceed along this path in

A1, *T*symb is accepted. Since A1 is just a slight adaptation of the word automaton recognizing the symbolic models of a linear time formula, we can easily conclude that ⟨*T*symb*, π*⟩ |=symb *ψ* and *T*symb |=symb *φ* by using the correspondence between the maximal subformulas.

# Construction of A*r*

abs

Let SV0(*φ,* A) be the projection of SV(*φ,* A) on the set of variables obtained by keep- ing the relations referring to variables of the current state only: *X* ∈ SV0(*φ,* A) iff there exists *sv* ∈ SV(*φ,* A) s.t. for every *x*1*,... , xn* ∈ VAR we have R(*x*1*,... , xn*) ∈ *X* iff R(*x*1*,... , xn*) ∈ *sv* . As a consequence of this definition, ⟨SV0(*φ,* A)*, symb*0⟩ verifies (SV) where *symb*0 : (VAR → *D*) → SV(*φ*) is defined by *symb*0(*v*) = *X* iff *v* satisfies all the constraints of *X*.

We use the elements of SV0(*φ,* A) to abstract the current state of the automaton

A. We build from A a standard Bu¨chi automaton Aabs = ⟨*Q*'*,I*'*,F* '*, δ*'⟩ over the alphabet SV(*φ,* A) s.t. *Q*' = *Q* × SV0(*φ,* A) and the transition relation is defined

by ⟨*q, X*⟩ −→*sv* ⟨*q*'*,X*'⟩ iff:

* *X* is the projection of *sv* w.r.t *V* , i.e. R(*x*1*,... , xn*) ∈ *X* iff R(*x*1*,... , xn*) ∈ *sv* .
* *X*'[*x* ← X*x* | *x* ∈ VAR] is the projection of *sv* ' w.r.t. the terms of the form X*x* where *X*'[*x* ← X*x* | *x* ∈ VAR] is obtained from *X*' by substituting every occurrence of *x* by X*x* for every *x* ∈ *V* . This means that R(*x*1*,... , xn*) ∈ *X* iff R(X*x*1*,... ,* X*xn*) ∈ *sv* .
* there exists a transition *q* −→*α q*' in A such that *sv* |= *α*.

symb

The set of initial states is *I*' = {⟨*q*0*, symb*0(*v*0)⟩ | *q*0} where *v*0 is the initial valuation

and the set of final states *F* ' = *F* × SV0(*φ,* A). This construction can be done in exponential time if the satisfaction relation can be checked in exponential time. The construction implies a bisimulation between a state ⟨*q, v*⟩ of A and ⟨*q, X*⟩ of Aabs such that *symb*0(*v*)= *X*. We can establish a stronger result in order to have a correspondence between paths in the automata A and Aabs. According to the definition of Sect. [3](#_bookmark2), one-step consistency in the case of symbolic valuations, which is a particular case of frame, is given by: for every *i*1*,... , ik >* 0, R(X*i*1 *x*1*,... ,* X*ik xk*) ∈ *sv* 1 ⇔ R(X*i*1 −1*x*1*,... ,* X*ik* −1*xk*) ∈ *sv* 2. This definition is naturally extended for one- step consistent sequences of symbolic valuations.

Lemma E.1 *There is a path* ⟨*q ,v* ⟩ *α*−→0

⟨*q ,v* ⟩ *α*−→1

*... in* A *iff there exists a path*

*sv* 0

*sv* 1

0 0 1 1

⟨*q*0*, X*0⟩ −→ ⟨*q*1*, X*1⟩ −→ *... in* Aabs *such that σ* = *v*0 · *v*1 ··· *and ρ* = *sv* 0 ··· *sv* 1 ···

*verify*

* + *ρ is one-step consistent,*
  + *σ* |= *ρ, i.e. for every i* ∈ N *we have σ, i* |= *ρ*(*i*)*. In particular for every i* ∈ N *we have vi* |= *Xi.*

Proof Let ⟨*q*0*, v*0⟩

*α*

−→ ⟨*q*1*, v*1⟩

0

*α*

−→ *...* be a path in A. By construction of the

1

*αi αi*+*l*−1

abstraction, there exists for each subpath ⟨*qi, vi*⟩ −→ ··· −−−→ ⟨*qi*+*l, vi*+*l*⟩ of length *l*

*sv* (*vl*)

a transition in Aabs of the form ⟨*qi, Xi*⟩ −−−→*i* ⟨*qi*+1*, Xi*+1⟩ where *vl* is the valuation

*i*

defined by *vl*(X*jx*)= *vi*+*j*(*x*) for all *j* ∈ {0*,... , l*}. Indeed, by the conditions (SV) on symbolic valuations and the definition of the symbolic satisfaction relation we have

*i*

* + if *vi* satisfies all the constraints of *Xi* then it is also the case for *vl* and so *sv* (*vl*)

*i* *i*

symbolically satisfies all the constraints of *Xi*,

* + similarly, if *vi*+1 satisfies all the constraints of *Xi*+1 then *sv* (*vl*) symbolically

*i*

satisfies the constraints of *X*'[*x* ← X*x* | *x* ∈ VAR],

* + for every *j* ∈ {0*,... ,l* − 1}, if *vi* |= *αi* then *symb*(*vl*) |=symb X*iαi*.

*i*

We pose *sv i* = *symb*(*vl*) for every *i* ∈ N. Since the symbolic valuations *sv i* and

*i*

*sv i*+1 are defined w.r.t. paths that overlap, it is obvious that ⟨*sv i, sv i*+1⟩ is one-

*sv* 0

*sv* 1

*sv* 2

step consistent. Thus, the run ⟨*q*0*, X*0⟩ −→ ⟨*q*1*, X*1⟩ −→ ⟨*q*2*, X*2⟩ −→ ··· is such that

*ρ* = *sv* 0 · *sv* 1 ··· is one-step consistent. Moreover, since for every *i* ∈ N we have

*sv i* = *sv* (*vl*), the sequence *σ* = *v*0 · *v*1 ··· satisfies *ρ*.

*i*

*sv* 0 *sv* 1

Conversely, suppose that there is an infinite path ⟨*q*0*, X*0⟩ −→ ⟨*q*1*, X*1⟩ −→ *...* in

Aabs such that the sequence *ρ* = *sv* 0 · *sv* 1 ··· is one-step consistent. One can build from this path a path in A satisfying the requirements. We proceed by induction on the position in the path.

By definition of the transition relation of A

*α*0 *α*1

*αl*−1

abs, there is a path *q*0 −→ *q*1 −→ ··· −−→

*ql* in A such that for every *i* ∈ {0*,... ,l*−1} we have *sv* 0 |=symb X*iαi*. Since *symb* is a surjective function, there exists a valuation *v* such that *symb*(*v*)= *sv* 0. By definition

*α*0 *α*1 *αl*−1

of the symbolic satisfaction relation the finite path ⟨*q*0*, v*0⟩ −→ ⟨*q*1*, v*1⟩ −→ ··· −−→

⟨*ql, vl*⟩ such that for every *j* ∈ {0*,... , l*} and *x* ∈ VAR we have *vj*(*x*)= *v*(X*jx*) is a

valid path in A. Indeed, the pair ⟨*vj, vj*+1⟩ satisfies *αj* for every *j* ∈ {0*,... ,l* − 1}.

*α*0 *α*1 *αi*−2+*l*

Now suppose that there exists a path of the form ⟨*q*0*, v*0⟩ −→ ⟨*q*1*, v*1⟩ −→ ··· −−−→

⟨*qi*−1+*l, vi*−1+*l*⟩ in A that satisfies all the symbolic valuations from *ρ* up to the position *i* − 1. Since the sequence *ρ* is one-step consistent, if *sv i*−1 |=symb X*jαi*−1+*j* for every *j* ∈ {0*,... ,l* − 1} then *sv i* |=symb X*jαi*+*j* for every *j* ∈ {0*,... ,l* − 2}.

Moreover, the valuation *vl*−1 defined by *vl*−1(X*jx*)= *vi*+*j*(*x*) for all *j* ∈ {0*,... ,l*−1}

*i* *i*

partially satisfy *sv i*.

We can claim that there exists a transition of the form *q*

*αi*+*l*−1 *i*+*l*−1 −−−→ *qi*+*l*

in A

such that *sv i* |=symb X*lαi*+*l*−1. Otherwise, since the sequence *ρ* is one-step consistent

there is no transition *q*

*αi*+*l*−1 *i*+*l*−1 −−−→ *qi*+*l*

such that *sv*

*i*+*l*−1

|=symb

*αi*+*l*−1

and so the

transition ⟨*q*

*i*+*l*−1

*, Xi*+*l*−1⟩

*sv i*+*l*−1

−−−−→ ⟨*qi*+*l*

*, Xi*+*l*

⟩ does not exist in A

abs

. Since the

domain satisfy the completion property, there is a valuation *vi*+*l* extending *vl*−1

*i*

into *vl* such that *vl*(X*lx*) = *vi*+*l*(*x*) and *vl*

|= *sv i*. By definition of the symbolic

*i i* *i*

satisfaction relation, we have *vl* |= *αi*+*l*−1 and so the path can be extended by using

*i*

the transition ⟨*q ,v*

*αi*+*l*−1

*,v* ⟩.

*i*+*l*−1

*i*+*l*−1⟩ −−−→ ⟨*qi*+*l*

*i*+*l*

We now pose *d* to be the maximum of E(*φ*) + 1 and the maximal number of outgoing transition of states of A. We denote by Frame(*φ,* A) the set of frames of the form {1*,... , d*}*l* → SV(*φ,* A). A symbolic run of Aabs is a symbolic model *T*symb = ⟨*N,* {*n*0}*,* −→*,* Γsymb⟩ s.t.

* *N* ⊆ *Q* × SV0(*φ,* A) × {1*,... , d*}∗ where the last component describes the path from the root and distinguishes the copies of a same state of Aabs,
* Γsymb : *N* → Frame(*φ,* A),
* *n*0 is of the form ⟨*q*0*, X*0*, ϵ*⟩ s.t. ⟨*q*0*, X*0⟩ ∈ *I*' and Γsymb(*n*0)= *fr* 0 is s.t. *X*0 is the projection of *fr* 0(*w*) for all *w* ∈ {1*,... , d*}*l*,
* Let *n* = ⟨*q, X, wn*⟩ ∈ *N* be a node s.t. Γsymb(*n*)= *fr* and {*n*1*,... , nd*} its set of successors. We pose *nj* = ⟨*qj, Xj, wn* · *j*⟩ for every *j* ∈ {1*,... , d*}.

*sv j*

(A1) for all *j* ∈ {1*,... , d*} there is a transition ⟨*q, X*⟩ −→ ⟨*qj, Xj* ⟩ in Aabs and there

exists *w* ∈ {1*,... , d*}*l*−1 s.t. *fr* (*j* · *w*)= *sv j*,

*sv* ' ' ' '

(A2) for all ⟨*q, X*⟩ −→ ⟨*q ,X* ⟩ in Aabs there exist *j* ∈ {1*,... , d*} s.t. ⟨*q ,X* ⟩ = ⟨*qj, Xj* ⟩

and *w* ∈ {1*,... , d*}*l*−1 s.t. *fr* (*j* · *w*)= *sv j*.

(A3) for every *j* ∈ {1*,... , d*}, the pair ⟨*fr,* Γsymb(*nj*)⟩ is *j*-step consistent.

By construction we have the following property.

Lemma E.2 *For every path* ⟨*q*0*, X*0*, w*0⟩ −→ ⟨*q*1*, X*1*, w*1⟩ −→ ··· *in T*symb*, there exists*

*a run* ⟨*q ,X* ⟩ *s*−→*v* 0 *sv* 1 *... of* A *s.t. the sequence sv* · *sv* ··· *is one-step*

⟨*q ,X* ⟩ −→

0 0 1 1

*consistent.*

abs 0 1

Proof Consider a path *π* = ⟨*q*0*, X*0*, w*0⟩ → ⟨*q*1*, X*1*, w*1⟩ −→ ··· in *T*symb. By defini-

tion of *T*symb

= ⟨*N,* {*n*0}*,* −→*,* Γ

*sv*

symb⟩ there exists a transition ⟨*qi, Xi*⟩ −→ ⟨*qi*+1

*i*

→

*, Xi*+1⟩

in A*T*

for every *i* ∈ N, and so there is a path *q*

−*αi q*

*αi*

−−*i*+→1 *qi*

*i*

*i*+2

*α*

*i*

*i*+*l*−1 *i*

*α*

abs

in A such that

*i i*+1

*i*+2 −−→ ··· −−−→ *qi*+*l*

* *Xi* is the projection of *sv i* w.r.t *V* ,
* *Xi*+1[*x* ← X*x* | *x* ∈ *V* ] is the projection of *sv i* w.r.t. the terms of the form X*x*,
* *sv i* |=symb *αi*,

and a sequence *w*'' ∈ {1*,... , d*}*l* s.t. *fr i*(*w*'')= *sv i* where *fr i* = Γsymb(⟨*qi, Xi, i*⟩).

*i* *i*

We pose *w*' = *wπ*(*i*) ··· *wπ*(*i* + *l*) for all *i* ∈ N. By definition of frames, the

*i*

symbolic valuation *fr i*(*w*') coincide with *sv i* on the terms of the form *x* and X*x*. This implies that *Xi* is the projection of *fr i*(*w*'), *Xi*+1[*x* ← X*x* | *x* ∈ VAR] is

*i*

*i*

the projection of *fr i*(*w*') w.r.t. the terms of the form X*x*, and *fr i*(*w*') |=symb *αi*.

*i*

By iterating these arguments, we can prove that *fr i*+*j* (*w*'

*i*+*j*

*i*

) |=symb *αi*+*j* for every

*j* ∈ {1*,... , l*−1} and since by definition of the symbolic models the sequence *fr i*(*w*')·

*i*

*fr i*+1(*w*' ) ··· *fr i*+*l*−1(*w*' ) is one-step consistent we have *fr i*(*w*') |= X*jαi*+*j* for

*i*+1

*i*+*l*−1

*i*

all *j* ∈ {1*,... ,l* − 1}. So we have all the elements to show that there exists a

*fri*(*wi*) *T*

transition of the form ⟨*qi, Xi*⟩ −−−→ ⟨*qi*+1*, Xi*+1⟩ in Aabs. By induction, one can

*fr* (*w*' )

build a path ⟨*q*0*, X*0⟩ −−0 −→0

*fr* (*w*' )

⟨*q*1*, X*1⟩ −−1 −→1

· · · such that *fr*

0(*w*' ) · *fr*

1(*w*' ) ··· is

one-step consistent.

0

1

So we build an alternating tree automaton A*r*

abs

recognizing the symbolic models

corresponding to Aabs. The set of locations of A*r*

abs

is the same than the set of

locations of Aabs, as well as the sets of initial and final sates. The alphabet of

*r*

A

abs

is of course Frame(*φ,* A) and the transition relation is the translation of the

conditions defined for the construction of symbolic runs

*δ*(⟨*q, X*⟩*, fr* )=

sv

⟨*q,X*⟩ '

' *i*:∃*w,fr* (*iw*)=*sv*

(*i,* ⟨*q*'*,X*'⟩)

−→⟨*q ,X* ⟩

∧

(*i,* ⟨*q*'*,X*'⟩)*.*

*i*∈{0*,...,d*}

sv

⟨*q,X*⟩ ' '

−→⟨*q ,X* ⟩:∃*w,fr* (*iw*)=*sv*

The first part express that every transition of Aabs corresponds to a successor and the second part that every successor correspond to a transition of Aabs (in the case where E(*φ*) + 1 is greater than the maximal number of outgoing transition of states of A). Since this automaton will be intersected with A*φ* recognizing the set of symbolic models satisfying *φ*, we do not need to check one-step consistency (A3).