An automaton approach to Kripke semantics

Gaspard Douady

An automaton approach to Kripke semantics

Gaspard Douady

June 6, 2013

Abstract

We study the satisfiability problem over axiomatic extensions of modal logic K, with an automata-theoretic point of view. We exhibit optimal subclasses of alternating tree automata, thus providing a new framework for the satisfiability problem with frame constraints. Additionally, we clarify the correspondence between runs of automata and branches of the standard tableau construction for modal logic. We are confident this pioneer work is a milestone in bridging philosophers, logicians and computer scientists know-hows.

Contents

Introduction

Modal logic [2] is a formal logic that extends classical propositional and predicate logic to include operators expressing modality. It is general enough to capture many concepts such as knowledge, times, obligations ... Therefore, it has been extensively studied for decades. Modal logic can be embedded into First Order Logic while still having satisfactory expressive power together with good computational properties. In particular, for many modal logics the satisfiability problem (namely, the decision problem "Does a given formula have a model?") is decidable. Noticeably, the satisfiability problem is central as many practical problems reduce to it: constraints solving, consistency of logical statements, synthesis of program skeletons, etc. As a consequence many research communities developed effective methods to decide this problem. Essentially, there are two main approaches to solve satisfiability: *tableaux based methods* [1] and *automata-based methods* [8]. While the former comes from philosophers and logicians, the latter has been developed by computer scientists based on the formal language background.

It is worth noticing that computer scientists essentially interpret modalities as temporal ones, in order to capture execution sequences of computing systems. Importantly, temporal logic specifications often require the use of fix-point operators in order to express, e.g. fairness properties of distributed systems. Also, temporal logics are not aimed to distinguishing between a state-transition-based specification of a system and its unfolding into a (computation) tree. So that the satisfiability problem of temporal logics naturally reduces to the non-emptiness problem of tree automata.

On the other hand, philosophers and logicians deal with several varieties of modalities (knowledge, belief, plausibility, etc). Therefore, they need to consider additional constraints to capture the full meaning of these modalities: for instance, in epistemic logic, they often make the hypothesis that

"I know φ implies that I know that I know φ" ($\Box\varphi \Rightarrow \Box\Box\varphi$); the models to be considered for such logics must therefore rely on particular *frames*, in our example those frames whose accessibility relation is transitive (Axiom 4 of e.g. [2]). Tableau methods are proof systems that have been proved to be flexible enough to capture various constraints on frames: reflexivity (Axiom T), symmetry (Axiom B), seriality (Axiom D), euclideanity (Axiom 5), etc. However, such hypothesis on the frames shapes may prevent tree unfoldings to be legitimate models, so that it is *a priori* unclear how to exploit automata-based techniques for such settings.

The present contribution is an attempt to tune tree automata so that satisfiability problems over particular classes of frames reduce to the non-emptiness problem of such automata.

More precisely, we exhibit adequate subclasses of tree automata which enable one to solve the satisfiability problem for the several axiomatic extensions of the basic modal logic K: we have namely addressed the problem for KT (restriction to frames satisfying T), KD (restriction to frames satisfying D), and K4 (restriction to frames satisfying 4). Basically, we introduce two subclasses of alternating automata, respectively called *K-automata* and *simple-loop automata*. We then establish two main results. As a first result, we show that the non-emptiness problem for the two proposed classes of automata is in PSPACE. As a second result, and as previously announced, we describe, given an axiomatic extension $\mathcal{L}$, how to effectively transform a formula φ into an automaton $A_\varphi^\mathcal{L}$, so that φ is $\mathcal{L}$-satisfiable if, and only if, $A_\varphi^\mathcal{L}$ accepts some tree. Additionally, a model of φ can be derived from an accepted tree. The proof of this second result relies on establishing a correspondence between runs of automata and branches of tableau. As a corollary of these two results, the non-emptiness problem for *K-automata* and *simple-loop automata* is PSPACE-complete, just as the satisfiability problem of the considered axiomatic extensions of K is. Hence, the proposed classes of automata are essentially optimal.

The report is organized as follows. In section 1, we present the modal logic K and its axiomatic extensions KT, KD, and K4. In section 2, we present the satisfiability problem of the considered logics. and recalls tableau methods in section 3. In section 4, we recall alternating tree automata. The second part of the report is our contribution. Section 5 presents two subclasses of automata, *K-automata* and *simple-loop automata*, and establishes that their non-emptiness problem is in PSPACE. We reduce satisfiability problems to automata non-emptiness problems in section 6.

Part I
State of the Art

1 Kripke Semantics

1.1 Logic K

The formulas of modal logics are built from a countable non-empty set of primitive propositions $AP = \{p_1, p_2, ...\}$, the classical connectives $\wedge$ ('and'), $\vee$ ('or'), $\neg$ ('not') and the non classical unary modal connectives $\Box$ ('box') and $\Diamond$ ('diamond'). The set $\mathcal{L}$ of formulas is defined by the following grammar where $p \in AP$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi$$

For convenience we add the formulas $\bot$ that denote the constant false formula $p_1 \wedge \neg p_1$ and $\top = \neg\bot$ a constant true formula.

We define $Closure(\varphi)$ as the set of all subformulas of φ and their negations.

To give the semantics of modal logic we first define a *frame* $\langle W, R\rangle$ which consists of a non-empty set, W, called the set of possible worlds and a binary relation R, between worlds usually called the *accessibility relation*. For example for $u, v \in W$, $u\ R\ v$ we say that the world v is accessible from the world u. This gives us a pair $\langle W, R\rangle$ called a *frame*.

Next the frame is extended to a *model* by labeling each world with a set of propositions from the set of propositions AP, this specifies the *truth value* of each proposition in each world. We do so by defining a labeling function $V : W \to 2^{AP}$ such that $p \in V(u)$ if and only if p is *true* in the possible world u. This gives us a triple $\langle W, R, V\rangle$ called a *model*.

Then we inductively define the the truth of a formula in world w of a model M (resumed in figure 1) :

- $M, w \models p$ iff $p \in V(w)$
- $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$
- $M, w \models (\varphi_1 \wedge \varphi_2)$ iff $M, w \models \varphi_1$ and $M, w \models \varphi_2$
- $M, w \models (\varphi_1 \vee \varphi_2)$ iff $M, w \models \varphi_1$ or $M, w \models \varphi_2$
- $M, w \models \Box\varphi$ iff for all $u \in W, w\ R\ u$ implies $M, u \models \varphi$

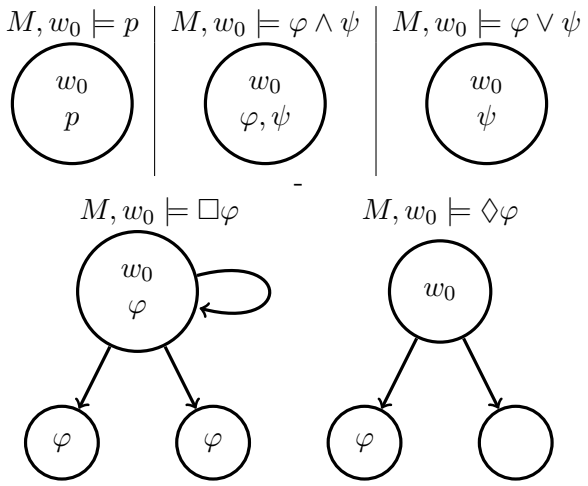


Figure 1: Satisfaction relation

- $M, w \models \Diamond\varphi$ iff there exists$u \in W, w\ R\ u$ and $M, u \models \varphi$

We read $M, w \models \varphi$ as w $satisfies$ φ and $\models$ is called the *satisfaction relation*. We say that a formula φ is valid in:

- a model $M = \langle W, R, V\rangle$ if and only if for all $w \in W,\ M, w \models \varphi$;
- a frame $F = \langle W, R\rangle$ if and only if for all$V$ φ is valid in $\langle W, R, V\rangle$

We say that two formulas φ and ψ are equivalent, written $\varphi \equiv \psi$, if, and only if, for allM, w we have $M, w \models \varphi \leftrightarrow M, w \models \psi$

In the following we will consider only formula in a *positive normal form* where negations only appear in front of atomic propositions. All formulas can be transformed into *negation normal form* by using the classic De Morgan's laws and the equivalence $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

1.2 Classes of frames

A *frame* is no more than a pointed graph, we now define *class of frames* $\mathcal{C}$ which is a set of frames such that every element in $\mathcal{C}$ satisfy a particular constraint over its accessibility relation or alternatively by a formula A (called *axiom*) that elements of this class should satisfy.

Name	Axiom	Frame condition
D	$\Box\varphi \to \Diamond\varphi$	serial: $\forall u \exists v, (u\ R\ v)$
T	$\Box\varphi \to \varphi$	reflexive: $\forall u\ (u\ R\ u)$
4	$\Box\varphi \to \Box\Box\varphi$	transitive: $\forall u, v, w (u\ R\ v) \wedge (v\ R\ w) \to (u\ R\ w)$

Table 1: Common axioms and corresponding conditions on relations

Name	Axioms	accessibility relation
K	-	All frames
KD	D	serial
K4	4	transitive
KT	T	reflexive

Table 2: Considered logics

In this report we focus on the set of constraint {D, T, 4} defined in table 1 and the following set of logic {K, KD, K4, KT} where each one is defined in table 2. we shall denote by $\mathcal{L}$ a typical element of this set. We can now define new logics by the restriction of the logic K to a certain class of frame. The table 2 contains all the logics that are considered in this report.

2 Decision Problems

We now present two major decision problems in logic.

2.1 Definition

- The *model checking problem* is: given a model M, a world $w \in M$ and a formula φ; decide whether of not $M, w \models \varphi$
- The *satisfiability problem* (and its dual the *validity problem*) is: given a logic $\mathcal{L}$ and a formula φ decide whether or not there exists a model $M = \langle W, R, V \rangle$ and a world $w \in W$ such that $M, w \models \varphi$ and the frame $F = \langle W, R \rangle$ is in the class of frames associated with $\mathcal{L}$. Such a formula is said to be $\mathcal{L}$-satisfiable.

For all considered logics the model checking problem is in P. Indeed adding constraints on the frames can't change the class of complexity of the model checking problem.

However the satisfiability problem of K is PSPACE-complete and adding constraints to the set of possible frames may create change the complexity of the satisfiability problem.

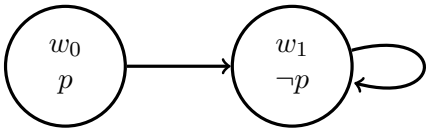
2.2 Example

$$\varphi_{ex} = p \wedge \Diamond\Box\neg p$$

.

Let us show that φ_{ex} is satisfiable in the logic K. We define *modele*:

$$M_{ex} = \langle W = \{w_0, w_1\}, R = \{(w_0w_1), (w_1w_1)\}, V(w_0) = \{p\}; V(w_1) = \emptyset\rangle$$



- $M_{ex}, w_0 \models \varphi_{ex}$ iff $M_{ex}, w_0 \models p$ and $M_{ex}, w_0 \models \Diamond\Box\neg p$
  - $M_{ex}, w_0 \models p$ iff $p \in V(w_0) \rightarrow$ True
  - $M_{ex}, w_0 \models \Diamond\Box\neg p$ iff $\exists w \in W, w_0\ R\ w \wedge M, w \models \Box\neg p$
    w_1 is the only successor of w_0, $M, w_1 \models \Box\neg p$ iff $\forall w \in W, w_1\ R\ w \Rightarrow M, w \models \neg p$
    w_1 is the only successor of w_1, $M, w_1 \models \neg p$ iff $p \notin V(w_1) \rightarrow$ True

We have $M_{ex}, w_0 \models \varphi_{ex}$ so φ_{ex} is satisfiable in the logic K

3 Tableau Methods

Tableau methods are proof systems that can be used as algorithm for solving the satisfiability problem. In this section, we present tableau methods for the modal logics previously defined. Then we demonstrate the soundness and the completeness of those methods, and finally we analyze the associated worst case complexities.

3.1 An Introductory Example

Let us show intuitively how to decide whether a modal formula is satisfiable or not. For instance let us consider the following formula:

$$\varphi_{ex} = \Box(\neg p \vee q) \wedge \Box p \wedge \Diamond\neg q.$$

We are going to try to create form scratch a model that satisfies this formula. In order to do so we decompose the formula as much as can be, use the truth conditions and infer constraints over our model.

- The initial world of our structure is called q_0 and it must satisfy φ_{ex}.
- The formula φ_{ex} is a conjunction hence q_0 must satisfy the three conjuncts $\Box(\neg p \vee q)$, $\Box p$ and $\Diamond \neg q$.
- As q_0 must satisfy $\Diamond \neg q$ we have to create a world q_1 such that $q_0 R q_1$ and q_1 must satisfy $\neg q$
- As $q_0 R q_1$ and q_0 must satisfy $\Box(\neg p \vee q)$ and $\Box p$, q_1 must satisfy $(\neg p \vee q)$ and p.
- As q_1 must satisfy $\neg p \vee q$ we consider two eventualities:
  - q_1 must satisfy $\neg p$, but q_1 must satisfy p we come to a contradiction.
  - q_1 must satisfy q, but q_1 must satisfy $\neg q$ we come to a contradiction.

We could not move forward as contradictions appeared in all possible scenarios. Hence we could not construct a model that satisfy φ_{ex}. In the following we formalize this model construction and prove both the soundness and completeness of this method. Therefore we proved that φ_{ex} is not satisfiable.

In the next section, we formalize this method called *tableau method*

3.2 Definition

In the introduction example, of previous section we decomposed the formula using an implicit set of rewriting rules. Such a set of rules is called a *tableau method*. The rules follow the same pattern as the definition of the truth conditions.

In the following rules these conventions stand:

- X and Y denote a finite possibly empty set of formulas
- φ and ψ denote formulas
- $\Box X$ stands for $\{\Box\varphi | \varphi \in X\}$

Definition 1 (Labeled formula)**.** *A* labeled formula *is a pair* $\langle n, \varphi \rangle$ *such that* $n \in \mathbb{N}$ *represent a world and* φ *is a formula. Labeled formulas are meant to say that the world* n *must satisfy the formula* φ*.*

Definition 2 (Branch)**.** *A* branch *is a pair of the form* $\langle L, F \rangle$ *such that* L *is a set of labeled formulas and* F *is a frame that basically characterize the transition relation between the world used in* L*. The branches are used to describe the constraints over the model that satisfies our initial formula.*

We say that a branch is closed *if* L *contains two labeled formulas of the form* $\langle n, \varphi \rangle$ *and* $\langle n, \neg\varphi \rangle$*. Intuitively a closed branch describes a contradictory set of constraint and means that there is no model that satisfy all the labeled formulas. On the other hand, a branch which can't be extended using any rules (the branch is said* saturated*) and is not closed (the branch is said* open*) then the initial formula is satisfiable in the frame describes by* F*.*

Definition 3 (Tableau)**.** *A* tableau *is a set of branches inductively defined by a set of rules and the formula we want to satisfy.*

The initial tableau T_φ *for a formula* φ *is the single branch* $\{\langle\{\langle 0, \varphi\rangle\}, \emptyset\rangle\}$

We say that a tableau is closed if all branches of the tableau are closed. A closed tableau built from an initial tableau T_φ *proves that* φ *is not satisfiable.*

Definition 4 (Rule)**.** *A* rule *(*ρ*) consists of a numerator* $\mathcal{N}$ *and a finite list of denominators* $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_k$ *separated by vertical bars:*

$$(\rho)\frac{\mathcal{N}}{\mathcal{D}_1|\mathcal{D}_2|...|\mathcal{D}_k}$$

The numerator and denominators are sets that may contain labeled formulas and pairs of states. A rule is applied to a branch B *that contains an instance* N *of* $\mathcal{N}$ *and replaces* B *by* k *branches* $B_i = B \bigcup D_i$ *where* D_i *is an instance of* $\mathcal{D}_i$ *with the same substitution than for* N*. We note that the rule can't be applied if one of the* D_i *is already included in* B*, that is if the rule as been applied before.*

3.3 Rules for K

Figure 2 defines the set of rules R_K for the logic K. We now explain the intuition behind these rules. The first two rules ($\wedge$) and ($\vee$) are enough to deal with the propositional calculus and the two other rules ($\Diamond$) and ($\Box$) deal with the relations between the worlds of a model.

- ($\wedge$): If the current world must satisfy $\varphi_1 \wedge \varphi_2$ then we know that it must satisfy both φ_1 and φ_2.

$$(\wedge)\frac{\langle n, \varphi_1 \wedge \varphi_2\rangle}{\langle n, \varphi_1\rangle, \langle n, \varphi_2\rangle} \qquad (\vee)\frac{\langle n, \varphi_1 \vee \varphi_2\rangle}{\langle n, \varphi_1\rangle | \langle n, \varphi_2\rangle}$$
$$(\Diamond)\frac{\langle n, \Diamond\varphi\rangle}{\langle m, \varphi\rangle : (n, m)} \qquad (\Box)\frac{\langle n, \Box\varphi\rangle : (n, m)}{\langle m, \varphi\rangle}$$

Figure 2: Set of rules R_K

	$\langle 0, \Box(\neg p \vee q) \wedge \Box p \wedge \Diamond\neg q\rangle$	
$(\wedge)$	$\langle 0, \Box(\neg p \vee q)\rangle, \langle \Box p\rangle, \langle 0, \Diamond\neg q\rangle$	
$(\Diamond)$	$\langle 1, \neg q\rangle, (0, 1)$	
$(\Box)$	$\langle 1, p\rangle$	
$(\Box)$	$\langle 1, \neg p \vee q\rangle$	
$(\vee)$	$\langle 1, \neg p\rangle$	$\langle 1, q\rangle$
	open	close

Table 3: Tableau of φ_{ex}

- $(\vee)$: If the current world must satisfy $\varphi \vee \psi$ then we know that either it must satisfy φ or *psi* so we create two branch that try the two alternatives.
- $(\Diamond)$: If the current world n must satisfy $\Diamond\varphi$ then we know that there has to be a successor m of n, that satisfies φ.
- $(\Box)$: If the current world n must satisfy $\Box\varphi$ then we know that all successors m of n must satisfy φ.

From a labeled formula $\langle n, \varphi\rangle$ the application of the tableau rules gives the minimal constraints that a model satisfying φ must verify.

Proposition 1. *A formula φ is K-satisfiable if, and only if, the tableau of φ from the set of rules R_K is not closed.*

The proof can be found in [4].

3.3.1 Example

Figure 3 gives the tableau for $\varphi_{ex} = \Box(\neg p \vee q) \wedge \Box p \wedge \Diamond\neg q$. We see that the tableau follows the demonstration previously done that φ is K-satisfiable.

$\langle 0, \Box\Diamond\top \wedge \Diamond\top\rangle$
$(\wedge) : \langle 0, \Box\Diamond\top\rangle, \langle 0, \Diamond\top\rangle$
$(\Diamond) : \langle 1, \top\rangle, (0, 1)$
$(\Box) : \langle 1, \Diamond\top\rangle$
$(\Box_4) : \langle 1, \Box\Diamond\top\rangle$
$(\Diamond) : \langle 2, \top\rangle, (1, 2)$
$(\Box) : ...$

Figure 3: Tableau K4 for $\Box\Diamond\top \wedge \Diamond\top$

3.4 Rules for Frames Constraints

In this section we present the rules for the three axioms {T,D,4} defined in tableau 4. These rules simulate constraints induced by the axioms over frames.

Table 4: Rules for axioms {T,D,4}

T	$(\Box_T)\dfrac{\langle n, \Box\varphi\rangle}{\langle n, \varphi\rangle}$
D	$(\Box_D)\dfrac{\langle n, \Box\varphi\rangle}{(n, m)}$
4	$(\Box_4)\dfrac{\langle n, \Box\varphi\rangle : (n, m)}{\langle m, \Box\varphi\rangle}$

- $(\Box_T)$ simulates reflexivity: we add φ in any node that contains $\Box\varphi$
- $(\Box_D)$ simulate seriality: we add a successor to any node containing a formula $\Box\varphi$. We note that we do not need to add a successor for nodes that do not contain any $\Box\varphi$ as we can safely add a loop to such kind of nodes.
- $(\Box_4)$ simulate transitivity: we add $\Box\varphi$ to any successor of a node containing $\Box\varphi$. We note that this rule does verify the subformula property thus the tableaux for K4 may be infinite. (example in figure 3)

The following propositions give the tableau methods for the considered logics {KT, KD, K4}.

Proposition 2. *A formula φ is KT-satisfiable if, and only if, the tableau of φ obtained by the set of rules $R_K \cup \{(\Box_T)\}$.*

Proposition 3. *A formula φ is KD-satisfiable if, and only if, the tableau of φ obtained by the set of rules $R_K \cup \{(\Box_D)\}$.*

Proposition 4. *A formula φ is K4-satisfiable if, and only if, the tableau of φ obtained by the set of rules $R_K \cup \{(\Box_4)\}$.*

The proof can be found in [4].

4 Generalities on Alternating Tree Automata

4.1 Alternating tree automata

In this section we find a definition of alternating automata slightly different of the classic definition [3], in particular the transition relations has been modified to a more suitable definition for our problem.

4.1.1 Preliminary Definitions

As our automata take infinite tree in input we need to give a formal definition of trees.

Definition 5. *A tree is a set $T \subseteq \mathbb{N}^*$ such that $\varepsilon \in T$ and if $x.c \in T$ where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then $x \in T$ and $\forall 0 \leq c' < c,\ x.c' \in T$.*

- *The elements of T are called* nodes.
- *The empty word ε is the* root *of the tree T.*
- *For every $x \in T$, the nodes $x.c$ where $x.c \in T$ are the* successors *of x.*
- *The number of successors of x is called the* degree *of x and is denoted $d(x)$.*
- *A node x is a* leaf *if $d(x) = 0$.*
- *A* path *π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x.c \in T$.*
- *The height $h(T)$ of a finite tree T is defined as the maximum of the lengths of words in T, (for instance $h(\{\varepsilon\}) = 0$).*

A tree is a bounded degree tree *if* $sup\{d(x)\}_{x \in T}$ *is finite.*

In the rest of the report, we will denote by $T|x$ the subtree of T at position x.

Given an alphabet Σ, a Σ-*labeled tree* is a pair $\langle T, V \rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ.

For a given set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$ where we also allow the formulas $\top$ and $\bot$). We will denote by $\bowtie$ any of the two Boolean operators $\wedge$ and $\vee$.

A *valuation* is a subset $Y \subseteq X$. Given a formula $\Theta \in \mathcal{B}^+(X)$, we say that Y *satisfies* Θ, written $Y \models \Theta$, iff assigning $\top$ to all elements of Y and $\bot$ to all elements of $X \setminus Y$ makes Θ *true*.

4.1.2 Definition of Alternating Tree Automata

Automata over infinite trees (tree automata) run over 2^{AP}-labeled treethat have no leaves. We first refer to automata over infinite *fixed arity* trees (i.e when $\exists d \in \mathbb{N}, T = \{0, ..., d-1\}^*$), then extend the definition to bounded degree trees and allowing ε-transitions. The reader may note that the following definition of *alternating tree automata* have a lower expressiveness that those given in [6] as they are a dense version of a subclass.

Definition 6. *An* alternating tree automaton *is a structure* $\mathcal{A} = \langle AP, Q, k, \delta, q_0 \rangle$, *where* AP *is a set of variables,* Q *is a finite set of states, with an initial state* $q_0 \in Q$, *and* $\delta : Q \to 2^{AP} \times (\mathcal{B}^+(\{0, 1, ..., k-1\} \times Q)) \times (\mathcal{B}^+(\{0, ..., k-1\} \times Q))$ *is the transition function which maps a state* $q \in Q$ *to a triple* $\langle \sigma, \Theta_1, \Theta_2 \rangle$ *where* $\sigma \subset AP$ *and* Θ_1 *and* Θ_2 *are two Boolean formulas.*

When the automaton is in state q with $\delta(q) = \langle \sigma, \Theta_1, \Theta_2 \rangle$ while reading input node x labeled by the set of propositions $V(x) \subseteq AP$, then if $\sigma \subseteq V(x)$ (resp. $\sigma \not\subseteq V(x)$) it proceeds by choosing a set $\mu = \{(c_1, q_1), ..., (c_n, q_n)\} \in \{0, ..., k-1\} \times Q)$ such that $\mu \models \Theta_1$ (resp. $\mu \models \Theta_2$). This corresponds to the existential choice. Then by splitting into n copies such that the j^{th} copy enters state q_j and proceeds to the input node $x.c_j$ otherwise. This corresponds to the universal choice.

We now generalize alternating automata[1] to infinite and finite trees of bounded and variable degrees. Formally, an alternating automata is now of the form $\mathcal{A} = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ where $\mathcal{D} \subset \mathbb{N}$ specifies the set of possible degrees. As expected, the transition function is adapted accordingly by

[1] while keeping the same terminology

defining $\delta : (Q \times \mathcal{D}) \to 2^{AP} \times (\mathcal{B}^+(\mathbb{N} \times Q)) \times (\mathcal{B}^+(\mathbb{N} \times Q))$ with the requirement that for every $k \in \mathcal{D}$, we have $\delta(q,k) \in 2^{AP} \times (\mathcal{B}^+(\{0,1,...,k-1\} \times Q)) \times (\mathcal{B}^+(\{0,...,k-1\} \times Q))$. When the automaton is in state q and reads an input node of degree k, it proceeds according to transition $\delta(q,k)$. We define the size $|\delta|$ of the transition function δ as the sum of the lengths of the formulas $\delta(q,k)$, for all $q \in Q$ and $k \in \mathcal{D}$.

Finally, we add the ε-transitions to alternating tree automata. The signature of the transition function is changed to $(Q \times \mathcal{D}) \to 2^{AP} \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q)) \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q))$. Elements of the form (ε, q_i) mean that the i^{th} copy of the automata moves to state q_i and leave the input node unchanged.

The *size* of an alternating automaton $\mathcal{A} = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ is simply $|\mathcal{A}| := |AP| + |Q| + |\mathcal{D}| + |\delta|$.

We now define a *run* of an alternating automaton $\mathcal{A}$ over a tree $\langle T, V \rangle$ is a 2^Q-labeled tree $\langle T_r, r \rangle$ where a node x of T_r, labeled by Γ, describes a set of copies of the automaton in a state $q \in \Gamma$ with input node $x \in T$. We note that $T_r \subseteq T$ as one node of T correspond at most one node of T_r

Definition 7. *Formally, a run $\langle T_r, r \rangle$ is a 2^Q-labeled tree. And we say that a run $\langle T_r, r \rangle$ is accepting if it satisfies the following:*

- $q_0 \in r(\varepsilon)$.
- *Let $x \in T_r$ with $\forall q \in r(x), \delta(q, d(x)) = (\sigma^q, \Theta_1^q, \Theta_2^q)$. In order to satisfy the transition relation, there must be valuation $\mu = \{(c_0, q_0), (c_1, q_1), ..., (c_n, q_n)\} \subseteq \{\varepsilon, 0, ..., d(x)-1\} \times Q$ such that the following conditions hold:*
  - *For all $q \in r(x)$ if $\sigma^q \subseteq V(x)$ then μ has to satisfies Θ_1^q otherwise μ has to satisfies Θ_2^q.*
  - *For all $0 \leq i \leq n$, we have $c_i \in \{0, ..., d(x)-1\}$ implies that $x.c_i \in T_r$ and $q_i \in r(x.c_i)$*
  - *For all $0 \leq i \leq n$, we have $c_i = \varepsilon$ implies $q_i \in r(x)$.*

A tree $\langle T, V \rangle$ is *accepted* by an automaton $\mathcal{A}$ if and only if there exists an accepting run $\langle T_r, r \rangle$. We note that there may not be a run if a formula Θ can not be satisfied. We denote the language of A as $L(A)$ the set of all 2^{AP}-labeled trees that A accepts.

In this definition of automata we only consider safety conditions (simulated by unsatisfiable transitions) that constraint allowed us to use a succinct

definition of runs that would not be correct if we had used parity acceptance conditions.

The definition given in this paper does not add any expressiveness compared to the classic definition:

Theorem 1. *Given a automaton A as defined here there exists an automaton A' as defined in the literature such that $L(A) = L(A')$*

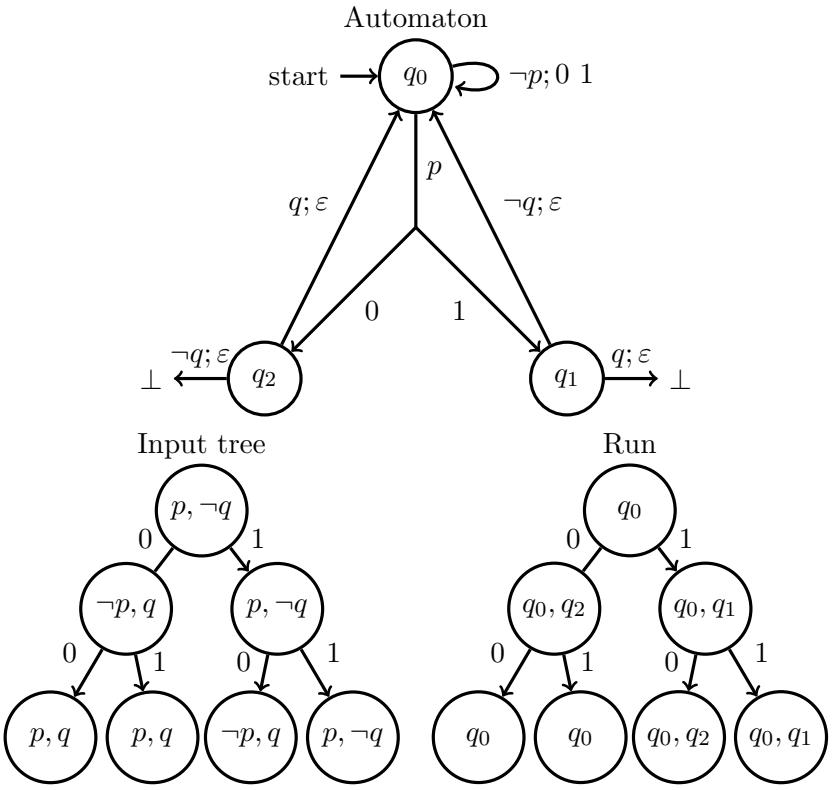
This has been formally proved yet.

4.2 Example

Here we give an example of an automaton A_{ex}, an accepted tree and the associated run. This automaton takes binary possibly infinite tree and ensures that whenever a node of the input tree contains p its left successor contains q and the right one does not.

$$A_{ex} = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$$

- $AP = \{q, p\}$
- $Q = \{q_0, q_1, q_2\}$
- $\mathcal{D} = \{0, 2\}$
- δ is such that:
  - $\delta(q_0, 2) = (\{p\}, (0, q_2) \wedge (1, q_1), (0, q_0) \wedge (1, q_0))$
  - $\delta(q_1, 2) = (\{q\}, \bot, (\varepsilon, q_0))$
  - $\delta(q_0, 2) = (\{q\}, (\varepsilon, q_0), \bot)$
  - $\delta(q_i, 0) = (\emptyset, \top, \top)$



4.3 Decision problem

Definition 8. *The* non-emptiness problem *is the following decision problem:*

Input: *An alternating automaton A.*

Output: *$L(A) \neq \emptyset$?*

The non-emptiness problem for alternating tree automata is EXPTIME-complete [3] that explain our need for PSPACE subclasses that could be used to reduce the *$\mathcal{L}$-satisfiability*.

In Part II, we explore PSPACE subclasses of alternating tree automata that can be used for solving the $\mathcal{L}$-satisfiability decision problem for the logics defined in figure 2.

Part II
Contribution

5 Subclasses of Alternating Tree Automata

In this section we introduce two subclasses of alternating automata, namely *K-automata* and *simple-loop automata*. Both classes have some features in common: states are partially ordered and transitions have only three types.

K-automata have not cycle between state while SL-automata may have self looping states.

5.1 K-automata (KA)

5.1.1 Definition

K-automata are defined as follows.

Definition 9. *A* K-automaton (KA) *is a structure* $A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ *where*

- *The set of states* Q *is equipped with a partial order* $\prec$ *over* Q*;*
- *The set of states is partitioned into* $Q = Q_\Diamond \bigsqcup Q_\Box \bigsqcup Q_b$ *where* $\bigsqcup$ *denote the disjoint union;*
- *The transition function* $\delta : (Q \times \mathcal{D}) \to 2^{AP} \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q)) \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q))$ *satisfies the following properties:*
  1. $\forall q_\Diamond \in Q_\Diamond, \exists q' \prec q_\Diamond, \exists p_0, ..., p_n \prec q_\Diamond, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:

$$\delta(q_\Diamond, k) = (\sigma, \bigvee_{i=0}^{k} (i, q') \wedge \Theta_1, \bigvee_{i=0}^{k} (i, q') \wedge \Theta_2)$$

  2. $\forall q_\Box \in Q_\Box, \exists q' \prec q_\Box, \exists p_0, ..., p_n \prec q_\Box, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:

$$\delta(q_\Box, k) = (\sigma, \bigwedge_{i=0}^{k} (i, q') \wedge \Theta_1, \bigwedge_{i=0}^{k} (i, q') \wedge \Theta_2)$$

3. $\forall q_b \in Q_b, \exists p_0, ..., p_n \prec q_b, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:

$$\delta(q, k) = (\sigma, \Theta_1, \Theta_2)$$

To ease the notations, given $\Gamma \subseteq Q$, we let $\Gamma_\Diamond := \Gamma \cap Q_\Diamond$ and $\Gamma_\Box := \Gamma \cap Q_\Box$.

For a state $q \in Q_\Diamond \cup Q_\Box$ we define *the successor of* q, as $Next(q)$ is the unique state such that $\delta(q, k) = (\sigma, \bowtie_{i=0}^{k} (i, q_{next}) \wedge \Theta_1, \bowtie_{i=0}^{k} (i, q_{next}) \wedge \Theta_2)$.

Finally, we extend the partial order $\prec \subseteq Q \times Q$ to $2^Q \times 2^Q$ by

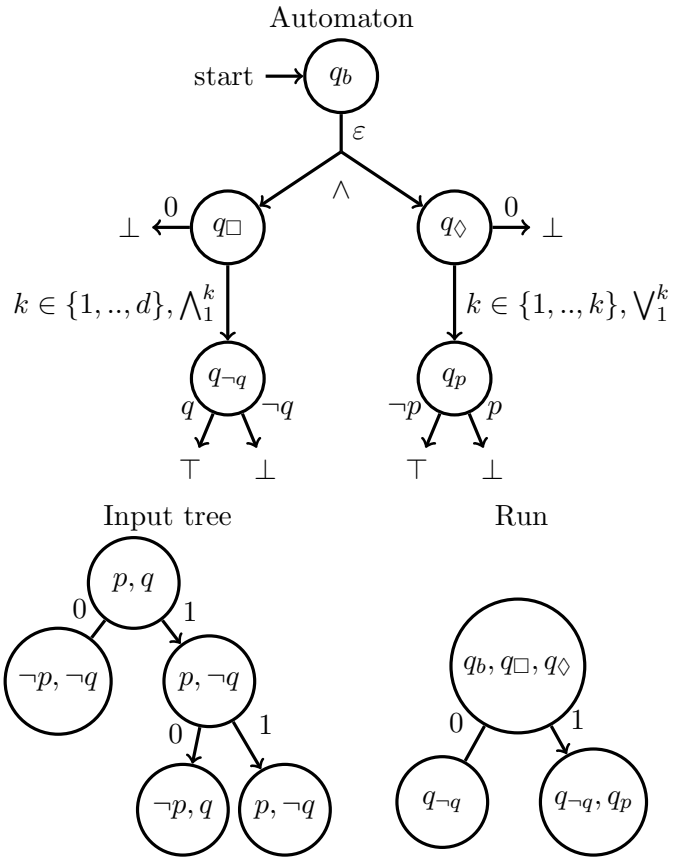
$$\Gamma \prec \Gamma' \text{ iff } \forall q \in \Gamma, \exists q' \in \Gamma', q \prec q'$$

5.1.2 Example of K-automaton

First we note that the example given in section 4 is not a K-automaton as the transition relation δ has a cycle. We give here an example of a K-automaton that accepts trees such that the root has at least one successor that contains p and all successors of the root must contain $\neg q$.

$$A_{ex} = \langle AP, Q, \mathcal{D}, \delta, q_b \rangle$$

- $AP = \{p, q\}$
- $Q_\Diamond = \{q_\Diamond\}$
- $Q_\Box = \{q_\Box\}$
- $Q_b = \{q_b, q_p, q_{\neg q}\}$
- $\mathcal{D} = \{0, ..., d\}$
- δ is such that:
  - $\delta(q_b, k) = (\emptyset, (\varepsilon, q_\Box) \wedge (\varepsilon, q_\Diamond), \bot)$
  - $\delta(q_\Diamond, k) = (\emptyset, \bigvee_{i=0}^{k-1}(i, q_p, \bot)$
  - $\delta(q_\Diamond, 0) = (\emptyset, \bot, \bot)$
  - $\delta(q_\Box, k) = (\emptyset, \bigwedge_{i=0}^{k-1}(i, q_{\neg q}, \bot)$
  - $\delta(q_\Box, 0) = (\emptyset, \top, \bot)$
  - $\delta(q_{\neg q}, k) = (\{q\}, \bot, \top)$
  - $\delta(q_p, k) = (\{p\}, \top, \bot)$



In section 6 we will see that this automaton corresponds to the formula $\varphi = \Diamond p \wedge \Box \neg q$.

5.1.3 Results on K-automata

Theorem 2. *Given a K-automaton A, if A accepts a tree, then A accepts a finite tree.*

Proof. Let $A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ We have $\forall q \in Q, \forall k \in \mathcal{D}$, the formulas of $\delta(q, k)$ only contain states strictly lower than q. Hence given an accepting run $\langle T_r, r \rangle$ for all $x \in T_r$ we can remove all states q of $r(x.i)$ that does not satisfy $\exists q' \in r(x), q \prec q'$ and still have an accepting run. Now let $r(x)$, when x range over the nodes of a branch of T_r, is strictly decreasing. So T_r is finite. □

Turning now to the main decision problem for KA, we can show the following statement.

Theorem 3. *The non-emptiness problem of the K-automata is in PSPACE.*

The proof of Theorem 3 is involved. The rest of this section is dedicated to its proof.

We will describe such an algorithm whose worst case complexity in space is polynomial. And prove its correctness. Here are the main ingredients to design the algorithm: note that proving the non-emptiness of an automaton A amounts to exhibiting an accepting run.

Our algorithm (see algorithm 2) performs a depth-first search along a non-deterministically on-the-fly guessed run and checks whether this run of A is accepting or not. As we will see, we will not need to keep the whole run in memory (the run is of size exponential in $|A|$), but only a branch of the run, which is of polynomial size.

Guessing the run involves guessing both the label σ and the degree k of each input node. We note that one major constraint of our subclass is the addition of the partial order $\prec$ over the set of state Q and the obligation of δ to point only to lower states; the latter means that all accepting run are finite trees.

We equip the algorithm with an auxiliary function $Saturate$ which given a degree k, a tree label σ, a set of state Γ, computes an "ε-closure" of Γ, that is a new set Γ' of all the states the automaton can reach by non-deterministically firing as many ε-transitions as possible. We note that this function may fail, for example if it tries to fire a transition labeled by the formula $\bot$.

The algorithms $Saturate$ and $NonEmptiness$ are given by Algorithm 1 and Algorithm 2 respectively.

Algorithm 1 $Saturate(A = \langle AP, Q, \mathcal{D}, \delta, q_0\rangle, k \in \mathcal{D}, \sigma \subseteq AP, \Gamma \subseteq Q)$

$\Gamma' := \Gamma$
$\Delta := \Gamma$
while $\Delta \neq \emptyset$ **do**
    $\Gamma_{aux} := \Gamma'$
    for $q \in \Delta$ **do**
        Let $(\sigma_q, \Theta_1^k, \Theta_2^k) = \delta(q, k)$
        if $\sigma_q \subseteq \sigma$ **then**
            $\Theta := \Theta_1^k$
        else
            $\Theta := \Theta_2^k$
        end if
        $(\exists)$ a valuation $\mu \subseteq \{\varepsilon, 0, ..., k\} \times Q$ such that $\{q|(c, q) \in \mu\} \prec \Gamma$
        if $\mu \models \Theta$ **then**
            $\Gamma' := \Gamma' \cup \{q'|(\varepsilon, q') \in \mu\}$
        else
            Reject
        end if
    end for
    $\Delta := \Gamma' \setminus \Gamma_{aux}$
end while
if $|\Gamma'_{\Diamond}| = k$ **then**
    Return Γ'
else
    Reject
end if

Algorithm 2 $NonEmptiness$ ($A = \langle AP, Q, \mathcal{D}, \delta, q_0\rangle$, $\Gamma \subseteq Q$)

$(\exists)\sigma \subseteq AP$ ▷ We guess which proposition are true in the current world.
$(\exists)k \in \mathcal{D}$ ▷ We guess the degree of the current world
$(\exists)\Gamma' := Saturate(A, k, \sigma, \Gamma)$ ▷ We pull the ε-transitions
for $q_{\Diamond} \in \Gamma'_{\Diamond}$ **do** ▷ For each state $q_{\Diamond}$ we create a new child that goes to the successor of $q_{\Diamond}$ and all the successors of elements of $\Gamma'_{\Box}$.
    Let $\Gamma'' = \{q''|\exists q_{\Box} \in \Gamma'_{\Box}, q'' = next(q_{\Box})\}$
    $NonEmptiness(A, \{next(q_{\Diamond})\} \bigcup \Gamma'')$
end for
Succeed

We now establish the correctness and complexity of Algorithm 2.

Lemma 1. *$NonEmptiness(A, \Gamma)$ terminates and uses a space in $poly(|A|)$.*

Proof. By induction over $\prec$, we prove that for any k and σ, $NonEmptiness(A, \Gamma)$ terminates :

- Basic case: we have $\Gamma = \emptyset$, immediately for any $Saturate(A, k, \sigma, \Gamma) = \emptyset$ and the algorithm ends.
- Inductive case: we have $\Gamma \neq \emptyset$.

  Let us consider a call to $NonEmptiness(A, \Gamma)$. We prove that $\{next(q)\} \cup \Gamma'' \prec \Gamma$. Let $\Gamma' = Saturate(A, k, \sigma, \Gamma)$ as defined in line 2. By definition of $Saturate$: for all $q' \in \Gamma' \setminus \Gamma$, there exists $q \in \Gamma$ such that $q \succ q'$. From the constraints over δ, for all $q'' \in \Gamma''$, there exists $q' \in \Gamma'$ such that $q' \succ q''$. Then for all $q'' \in \Gamma''$ there exists $q \in \Gamma$ such that $q'' \prec q$. Furthermore, there exits $q \in \Gamma$ such that $next(q^{\Diamond}) \prec q^{\Diamond} \preccurlyeq q$. Hence we have proved $\{q_{next}\} \cup \Gamma'' \prec \Gamma$, so we can apply the inductive hypothesis.

Therefore the recursive call tree of the Algorithm $NonEmptiness$ is at most of depth $|Q|$ and of finite degree (bounded by $Q_{\Diamond}$). So the algorithm uses a polynomial space: we only require to store local variable for each calls and the call stack for the backtracking.

□

The following proposition expresses the correctness and completeness of Algorithm $NonEmptiness$.

Proposition 5. *Let $A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ be a K-automata.*

$$L(A) \neq \emptyset \text{ if, and only if, } NonEmptiness(A, \{q_0\}) \text{ succeeds.}$$

Proof. In the following proof we extend the definition of KA such that the initial state q_0 can be replaced by a set Γ_0 of initial states. (This is obviously equivalent as we can replace this set by a state q_0 that points to this set with an ε transition).

(completeness) Let $A = \langle AP, Q, \mathcal{D}, \delta, \Gamma \rangle$ be a KA with $L(A) \neq \emptyset$. Let (T_r, r) be an accepting run over a tree $(T_K, V_K) \in L(A)$, with $r(\varepsilon) = \Gamma_{\varepsilon}$.

Let the property:

$$P(h) = \text{"For all node } x \in T_r \text{ with } h(x) = h, \text{ we have for all } \Gamma^p \subseteq r(x) \ , \\ NonEmptiness(A, \Gamma^p) \text{ succeeds."}$$

we show this property by induction on h.

- $P(0)$: $h(x) = 0$ for all subset $\Gamma^p \subseteq r(x)$ there is a non-deterministic execution of $Saturate(A, V_K(x), 0, \Gamma^p) \subseteq r(x)$. As $r(x)_\Diamond = \emptyset$ the algorithm terminates and succeeds.

- $P(h) \Rightarrow P(h+1)$: for all subset $\Gamma^p \subseteq r(x)$ there is a non-deterministic execution $\Gamma' = Saturate(A, V_K(x), 0, \Gamma^p) \subseteq r(x)$. And for all $q_\Diamond \in \Gamma'_\Diamond$ there is a successor x' of x such that $\{next(q_\Diamond)\} \bigcup \{q'' | \exists q_\Box \in \Gamma'_\Box, q'' = next(q_\Box)\} \subseteq \Gamma_{x'}$. As $h(x') = h$ we can apply the induction hypothesis $P(h)$ over the node x'.

As T_r is finite, $P(h(T_r))$ gives us the completeness.

(correctness) Let $A = \langle AP, Q, \mathcal{D}, \delta, \Gamma \rangle$ be a KA such that there exists a successful execution of $NonEmptiness(A, \Gamma)$.

$P(h) =$ "For all call of $NonEmptiness(A, \Gamma_x)$ of height h we have for all $\Gamma^p \subseteq \Gamma_x$ an accepting run of $A' = \langle AP, Q, D, \delta, \Gamma^p \rangle$."

We show this property by induction on h.

- $P(0)$: If the execution tree has height 0, necessarily $\Gamma'_\Diamond = Saturate(A, \sigma, k, \Gamma_p)_\Diamond = \emptyset$, and the run $\langle \{\varepsilon\}, r(\varepsilon) = \Gamma' \rangle$ is an accepting run of $A' = \langle AP, Q, D, \delta, \Gamma^p \rangle$

- $P(h+1)$: by induction hypothesis we have an accepting run $\langle T_i, r_i \rangle$ for each $A_i = \langle AP, Q, D, \delta, \{next(q_\Diamond)\} \bigcup \Gamma''_x \rangle$(4). Hence the run $\langle T_r, r \rangle$ constructed by the concatenation of $\langle \{\varepsilon\}, r(\varepsilon) = \Gamma'^p \rangle$ with the the runs $\langle T_i, r_i \rangle$ defined as $T_r = \{i.c | \exists c \in T_i\} \cup \{\varepsilon\}$, $r(\varepsilon) = \Gamma'^p$ and $r(i.c) = r_i(c)$ is an accepting run of $A' = \langle AP, Q, D, \delta, \Gamma^p \rangle$.

As the recursive call tree of NonEmptiness is finite and of height h, the property $P(h)$ gives us the correction of our algorithm. □

5.2 Simple Loop Automata (SLA)

Simple Loop Automata (SLA) are very similar to K-automata. They only differ by the constraint over the states of $Q_\Box$ that now point to themselves. They are used to express the the fact that $\Box$ and $\Box^*$ are equivalent in transitive models and therefore verify the axiom 4. Although because of this loop we lose the property of finite accepting tree and the proof for the PSPACE-membership is more difficult.

5.2.1 Definition

Definition 10. *A* Simple Loop Automaton *is a structure* $A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ *where*

- *The set of states* Q *is equipped with a partial order* $\prec$ *over* Q*;*
- *The set of states is partitioned into* $Q = Q_\Diamond \bigsqcup Q_\Box \bigsqcup Q_b$ *where* $\bigsqcup$ *denote the disjoint union;*
- *The transition function* $\delta : (Q \times \mathcal{D}) \to 2^{AP} \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q)) \times (\mathcal{B}^+((\mathcal{D} \cup \varepsilon) \times Q))$ *satisfies the following properties:*
  1. $\forall q_\Diamond \in Q_\Diamond, \exists q' \prec q_\Diamond, \exists p_0, ..., p_n \prec q_\Diamond, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:
  $$\delta(q_\Diamond, k) = (\sigma, \bigvee_{i=0}^{k} (i, q') \wedge \Theta_1, \bigvee_{i=0}^{k} (i, q') \wedge \Theta_2)$$
  2. $\forall q_\Box \in Q_\Box, \exists q' \prec q_\Box, \exists p_0, ..., p_n \prec q_\Box, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:
  $$\delta(q_\Box, k) = (\sigma, \bigwedge_{i=0}^{k} ((i, q') \wedge (i, q_\Box)) \wedge \Theta_1, \bigwedge_{i=0}^{k} ((i, q') \wedge (i, q_\Box)) \wedge \Theta_2)$$
  3. $\forall q_b \in Q_b, \exists p_0, ..., p_n \prec q_b, \exists \Theta_1, \Theta_2 \in \mathcal{B}^+((\varepsilon, p_0), ..., (\varepsilon, p_n)), \exists \sigma \subseteq AP, \forall k \in \mathcal{D}$:
  $$\delta(q, k) = (\sigma, \Theta_1, \Theta_2)$$

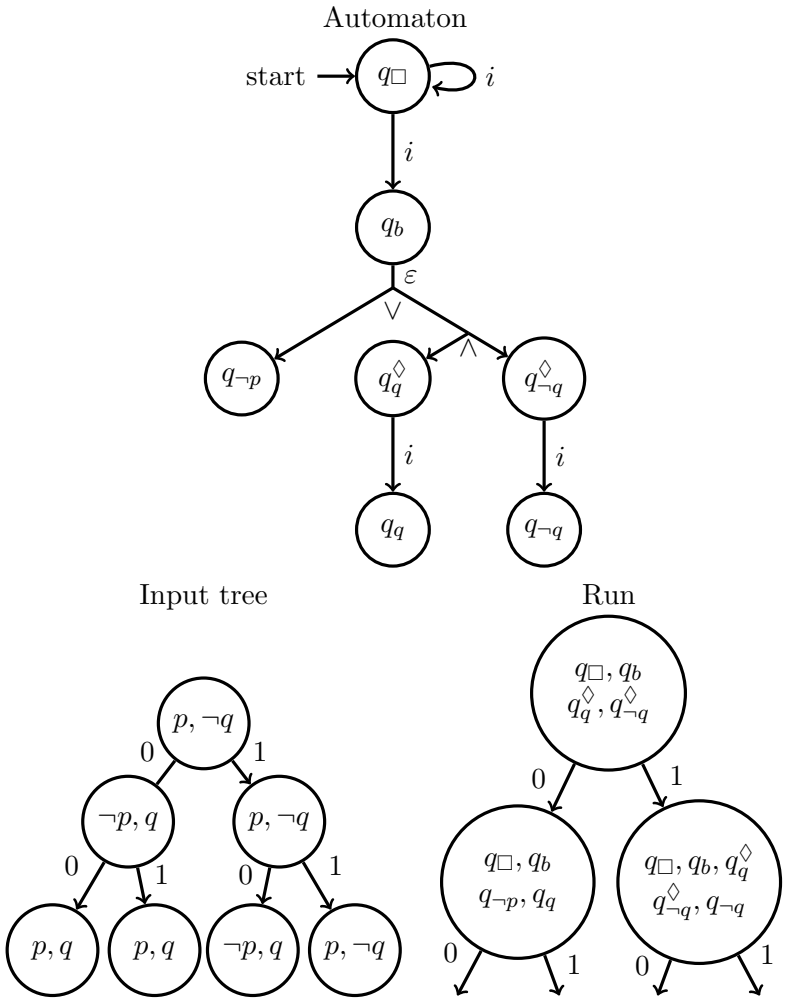
5.2.2 Example of SL-automaton

The automaton given in Section 4 is not a SLA. we can give here an example of an SLA that accepts a similar language (except that there is no difference between the left and right successors of a node). This gives that whenever a node contains p, one son must contain q and the other must contain $\neg q$.

$$A_{ex} = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$$

- $AP = \{q, p\}$
- $Q_\Box = \{q_\Box\}$
- $Q_\Diamond = \{q_q^\Diamond, q_{\neg q}^\Diamond\}$

- $Q_b = \{q_b, q_{\neg p}, q_q, q_{\neg q}\}$
- $\mathcal{D} = \{0, 2\}$
- δ is such that:
  - $\delta(q_\Box, 2) = (\emptyset, \bigwedge_{i=0}^{1}(i, q_b) \wedge (i, q_\Box), \bot)$
  - $\delta(q_b, 2) = (\emptyset, (\varepsilon, q_{\neg p}) \vee ((\varepsilon, q_q^\Diamond) \wedge (\varepsilon, q_{\neg q}^\Diamond)), \bot)$
  - $\delta(q_q^\Diamond, 2) = (\emptyset, \bigvee_{i=0}^{1}(i, q_q), \bot)$
  - $\delta(q_{\neg q}^\Diamond, 2) = (\emptyset, \bigvee_{i=0}^{1}(i, q_{\neg q}), \bot)$
  - $\delta(q_{\neg p}, k) = (\{p\}, \bot, \top)$
  - $\delta(q_q, k) = (\{q\}, \top, \bot)$
  - $\delta(q_{\neg q}, k) = (\{q\}, \bot, \top)$
  - $\delta(q \in Q_\Diamond \cup Q_\Box, 0) = (\emptyset, \top, \bot)$



5.2.3 Proofs

Theorem 4. *The non-emptiness problem of SL-automata is in PSPACE.*

The rest of this section is dedicated to the proof of Theorem 4: We follow the same pattern as the proof of the K-Automata, that is the design of an algorithm whose worst case complexity in space is polynomial.

We introduce an adaptation (algorithm 3) of the previous algorithm (algorithm 2) for the non emptiness problem. The skeleton is the same and it refers to same procedure Saturate (algorithm 1). Here the runs of SL-Automata are infinite but we claim that beyond a certain depth we can loop the branches on themselves. In order to find these loops we will keep in

memory in Λ the sequence of Γ's of the current branch and we will loop as soon as we see two identical Γ's in the parameter Λ.

The algorithms *Saturate* and *NonEmptiness* are given by Algorithm 1 and Algorithm 3 respectively.

Algorithm 3 $NonEmptinessSL$ ($A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$, $\Gamma \subseteq Q$, $\Lambda \subseteq 2^Q$)

1: $(\exists)\sigma \subseteq AP$ ▷ We guess which proposition are true in the current world.
2: $(\exists)k \in \mathcal{D}$ ▷ We guess the degree of the current world
3: $(\exists)\Gamma' := Saturate(A, k, \sigma, \Gamma)$ ▷ We pull the ε-transitions
4: **if** $\Gamma \in \Lambda$ **then**
5: Succeed
6: **end if**
7: **for** $q_\Diamond \in \Gamma'_\Diamond$ **do** ▷ For each state $q_\Diamond$ we create a new child that goes to the successor of $q_\Diamond$ and all the successors of elements of $\Gamma'_\Box$.
8: Let $\Gamma'' = \{q'' | \exists q_\Box \in \Gamma'_\Box, q'' = next(q_\Box)\} \cup \Gamma'_\Box$
9: $NonEmptinessSL(A, \{next(q_\Diamond)\} \cup \Gamma'', \Lambda \cup \Gamma)$
10: **end for**
11: Succeed

We now establish the correctness and complexity of Algorithm 3.

Lemma 2. *$NonEmptinessSL(A, \Gamma)$ terminates and uses a space in $poly(|A|)$.*

Proof. It easy to see that the sequence of Γ'''s called in line 7 are increasing and therefore there is at most $|Q_\Box|$ different Γ'''s. In the same way, there is at most $|Q_\Diamond|$ different $next(q_\Diamond)$. That leaves us at most $|Q_\Box| \times |Q_\Diamond|$ different $\{next(q_\Diamond)\} \cup \Gamma''$ on a branch of a recursive call tree of $NonEmptiness(\_)$.

Then from the test in line 3 the depth of a recursive call tree is at most $|Q_\Box| \times |Q_\Diamond|$ and of finite degree (bounded by $|Q_\Diamond|$). Hence the algorithm terminates and uses a polynomial space. □

The following proposition expresses the correctness and completeness of Algorithm *NonEmptiness*

Proposition 6. *Let $A = \langle AP, Q, \mathcal{D}, \delta, q_0 \rangle$ be a SL-automaton.*

$$L(A) \neq \emptyset \text{ if, and only if, } NonEmptiness(A, \{q_0\}, \emptyset) \text{ succeeds.}$$

Proof. ($\Rightarrow$) If $L(A) \neq \emptyset$ there is an accepting run of A and this run is a certificate for a successful execution of $NonEmptiness(A, \{q_0\}, \emptyset)$.

($\Leftarrow$) We only sketch the proof. From a successful execution of $NonEmptiness(A, \{q_0\}, \emptyset)$, we can build an embryo of an infinite run. To get an infinite run, we copy the subtree of node labeled by Γ to leaves labeled by the same Γ. □

6 From Formulas to Automata

In this section, we present several reductions: we show how each f $\mathcal{F}$-SAT decision problem (of Section 1) polynomially reduces to the non-emptiness problem of a particular class of automata (of Section 5). We first start with K-SAT, then quickly give the reduction for KD, KT and K4.

6.1 Reducing K-Sat to KA-emptiness

In this section, we prove the following result.

Theorem 5. *Given a K-formula φ we can construct in linear running time a K-automata A_φ such that φ is satisfiable if, and only if, $L(A_\varphi) \neq \emptyset$.*

Let φ be a K-formula. Each subformula of φ will give rise to a state of automaton A_φ, and the transition function δ is inspired from the rules of the tableau T_φ of φ (Section 3). More formally,

Definition 11. *for $\varphi \in K : A_\varphi = \langle AP, Q, \mathcal{D}, \delta, \varphi \rangle$ is defined by:*

- $AP =$ *the set of propositions that appear in* φ
- $Q = Closure(\varphi)$ *and*
  - $Q_\Box = \{\Box\psi | \Box\psi \in Closure(\varphi)\}$
  - $Q_\Diamond = \{\Diamond\psi | \Diamond\psi \in Closure(\varphi)\}$
  - $Q_b = Q \setminus (Q_\Box \cup Q_\Diamond)$
- *with* $\succ$ *as* "is a subformula of"
- $\mathcal{D} = \{0, ..., |Q_\Diamond|\}$
- φ *is the initial state*
- δ*:*
  - $\forall k \in \{1, ..., |Q_\Diamond|\}$*:*
    - $\forall p \in AP, \delta(p, k) = (\{p\}, \top, \bot)$
    - $\forall(\neg\psi) \in AP, \delta(\neg\psi, k) = (\{p\}, \bot, \top)$
    - $\delta(\varphi_1 \wedge \varphi_2, k) = (\emptyset, (\varepsilon, \varphi_1) \wedge (\varepsilon, \varphi_2), \bot)$
    - $\delta(\varphi_1 \vee \varphi_2, k) = (\emptyset, (\varepsilon, \varphi_1) \vee (\varepsilon, \varphi_2), \bot)$
    - $\delta(\Diamond\varphi_1, k) = (\emptyset, \bigvee_{i=0}^{k-1}(i, (\varphi_1)), \bot)$
    - $\delta(\Box\varphi_1, k) = (\emptyset, \bigwedge_{i=0}^{k-1}(i, (\varphi_1)), \bot)$

- $k = 0$:
  - $\forall p \in AP, \delta(p, 0) = (\{p\}, \top, \bot)$
  - $\forall p \in AP, \delta(\neg p, 0) = (\{p\}, \bot, \top)$
  - $\delta(\varphi_1 \wedge \varphi_2, 0) = (\emptyset, (\varepsilon, \varphi_1) \wedge (\varepsilon, \varphi_2), \bot)$
  - $\delta(\varphi_1 \vee \varphi_2, 0) = (\emptyset, (\varepsilon, \varphi_1) \vee (\varepsilon, \varphi_2), \bot)$
  - $\delta(\Diamond \varphi_1, 0) = (\emptyset, \bot, \bot)$
  - $\delta(\Box \varphi_1, 0) = (\emptyset, \top, \bot)$

Regarding complexity, it is easy to show that $|A_\varphi| \in O(|\varphi|)$.

The correctness of the construction of A_φ relies on a correspondence between accepting runs of A_φ and open branches of tableau T_φ for φ, which actually is an even stronger result. We develop this point in Section 6.1.2.

6.1.1 Example

$$\varphi_{ex} = \Diamond q \vee (\Box q \wedge \Diamond p)$$

$\langle 0, \Diamond q \vee (\Box q \wedge \Diamond p) \rangle$	
$(\vee) : \langle 0, \Diamond q \rangle$	$(\vee) : \langle 0, \Box q \wedge \Diamond p) \rangle$
$(\Diamond) : \langle 1, q \rangle, (0, 1)$	$(\wedge) : \langle 0, \Box q \rangle, \langle 0, \Diamond p \rangle$
	$(\Diamond) : \langle 1, p \rangle, (0, 1)$
	$(\Box) : \langle 1, q \rangle$

Figure 4: Tableau of φ_{ex}

this gives us the following automaton A_{ex}:

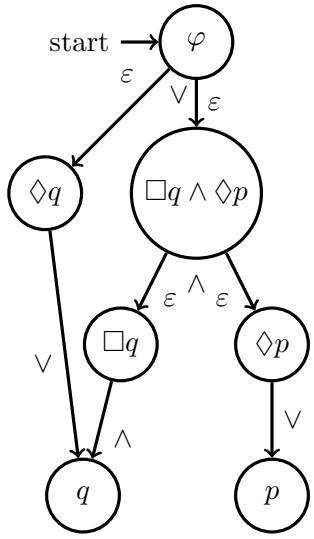
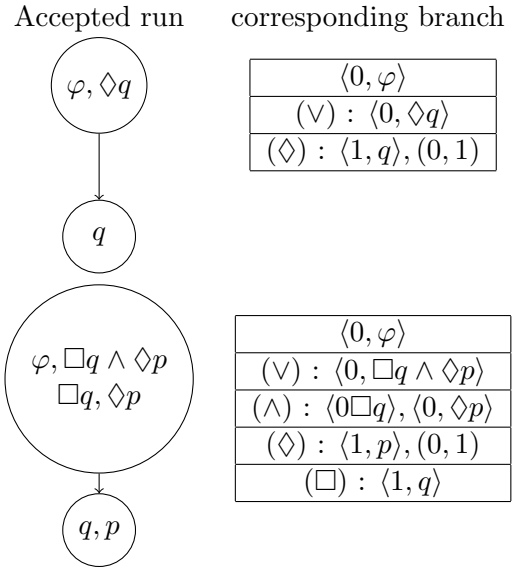


Figure 5: Automaton of φ_{ex}



6.1.2 Correction

Proposition 7. *$L(A_\varphi) \neq \emptyset$ iff there is an open branch in $\mathcal{T}_\varphi$.*

Proof. $\Rightarrow$) Let$\langle T_r, r \rangle$ be an accepting run over some tree $\langle T_K, V_K \rangle$ of A_φ, we

build an open branch of the tableau T_φ. By Theorem 2, we can assume that $\langle T_K, V_K \rangle$ is finite. We therefore can see T_K as a frame.

We define the branch (see Definition 2)

$$B_\varphi = \langle L, T_K \rangle$$

where the set of labeled formulas L are all the labeled formulas $\langle n, \varphi \rangle$ such that there exists a node $x \in T_r$ with $\varphi \in r(x)$.

Lemma 3. *B_φ is a branch of the tableau of $\mathcal{T}_\varphi$.*

Proof. We show by induction over the structure of φ that B_φ can be constructed following the rules of K from the initial branch $\{\langle\{\langle 0, \varphi\rangle\}, \emptyset\rangle\}$.

- $\varphi = p$: Immediate.
- $\varphi = \psi_1 \vee \psi_2$: From the definition of an accepting run and the definition of δ we have that the label of the root $r(\varepsilon) = (\varepsilon, \Gamma_0)$ is such that either ψ_1 or ψ_2 are in Γ_0. That is either the labeled formulas $\langle \varepsilon, \psi_1 \rangle$ or $\langle \varepsilon, \psi_2 \rangle$ are in B_φ. This is one of the two branch created by of the rule $(\vee) = \dfrac{\langle n, \varphi_1 \vee \varphi_2 \rangle}{\langle n, \varphi_1 \rangle \mid \langle n, \varphi_2 \rangle}$ over the initial branch.
- $\varphi = \psi_1 \wedge \psi_2$: From the definition of an accepting run and the definition of δ we have that ψ_1 and ψ_2 are in $r(\varepsilon)$. That is the labeled formulas $\langle \varepsilon, \psi_1 \rangle$ and $\langle \varepsilon, \psi_2 \rangle$ are in B_φ. This is the use of the rule $(\wedge) = \dfrac{\langle n, \varphi_1 \wedge \varphi_2 \rangle}{\langle n, \varphi_1 \rangle, \langle n, \varphi_2 \rangle}$ over the initial branch.
- $\varphi = \Diamond\psi$: From the definition of an accepting run and the definition of δ we have that there is $i \in \mathbb{N}$ such that $\psi \in r(i)$. That is the labeled formula $\langle i, \psi \rangle$ are in B_φ. This is the use of the rule $(\Diamond)$ over the initial branch.
- $\varphi = \Box\psi$: From the definition of an accepting run and the definition of δ we have that for all $i \in \mathbb{N}$ such that $i \in T_K$ we have $\psi \in r(i)$. That is the labeled formulas $\langle i, \psi \rangle$ are in B_φ. This is the use of the rule $(\Box)$ over the initial branch.

Hence B_φ is a branch of from the tableau of φ. □

Lemma 4. *B_φ is saturated and open.*

Proof. We proceed by contradiction. Assume a rule to create a new branch applies. We analyze such possible rules:

- Rule ($\vee$):
  - we can apply the rule ($\vee$)
  - there is a labeled formula $\langle x, \psi_1 \vee \psi_2 \rangle$,
  - the node x from the run is such that $\psi_1 \vee \psi_2 \in r(x)$
    - $\psi_1 \in r(x)$ so B_φ contains the labeled formula $\langle x, \psi_1 \rangle$
    - $\psi_2 \in r(x)$ so B_φ contains the labeled formula $\langle x, \psi_2 \rangle$
  - hence the contradiction.
- Rule ($\wedge$):
  - we can apply the rule ()
  - there is a labeled formula $\langle x, \psi_1 \vee \psi_2 \rangle$,
  - the node x in the run is such that $\psi_1 \wedge \psi_2 \in \Gamma_x$
  - $\psi_1 \in \Gamma_x$ and $\psi_2 \in \Gamma_x$
  - B_φ contains the labeled formulas $\langle x, \psi_1 \rangle$ and $\langle x, \psi_2 \rangle$
  - hence the contradiction.
- Rule ($\Diamond$):
  - we can apply the rule ($\Diamond$)
  - there is a labeled formula $\langle x, \Diamond psi \rangle$
  - the node x in the run is such that $\Diamond\psi \in r(x)$
  - there is a node $x.i$ in the run such $\psi \in r(x.i)$
  - B_φ contains the labeled formula $\langle x.i, \psi \rangle$
  - hence the contradiction.
- Rule ($\Box$):
  - we can apply the rule ($\Box$)
  - there is a labeled formula $\langle x, \Box psi \rangle$
  - the node x in the run is such that $\Box\psi \in \Gamma_y$
  - for all node $x.i$ in the run we have $\psi \in r(x.i)$
  - B_φ contains the labeled formulas $\langle x.i, \psi \rangle$

– hence the contradiction.

There is not any rules that can be applied to our branch, hence the branch is saturated.

Now let us assume that the B_φ is closed

- p and $\neg p$ are in the branch: immediate contradiction form the definition of an accepting run and δ.
- ψ and $\neg\psi$ are in the branch : no possible as φ is in a positive normal form.

Hence B_φ is an open branch of the tableau $\mathcal{T}_\varphi$. □

The Application of the Lemmas 3 and 4 concludes the first implication $\Rightarrow$).

$\Leftarrow$)

We now prove the completeness of our construction, namely that from an open branch of $T_\varphi = \langle L, T_K\rangle$ we can construct an accepting run $\langle T_K, r\rangle$ over a tree T_K, V_K of A_φ.

First we construct a run $\langle T_K, r\rangle$ over a tree $\langle T_K, V_K\rangle$:

- $V_K(x) = \{p | \langle x, p\rangle \in L\}$.
- $r(x) = \{q | \langle x, q\rangle \in L\}$)

We now show that $\langle T_K, r\rangle$ is indeed an accepting run of A_φ. Which is pretty easy by verifying that all the formulas are indeed satisfied. □

The main Theorem 5 is a simple corollary of the correction of tableau constructions(Section 3) and Proposition 7.

The object A_φ is central in this work. We will reuse it in further section for the cases KD and KT.

6.2 Reducing KD-Sat to KA-emptiness

In this section, we prove the following result.

Theorem 6. *Given a K-formula φ we can construct in linear running time a K-automata A^D_φ such that φ is KD-satisfiable if, and only if, $L(A_\varphi) \neq \emptyset$.*

Let φ be a K-formula. Each subformula of φ will give rise to a state of automaton A^D_φ, and the transition function δ is inspired from the tableau rules for KD (Section 3). More formally,

Definition 12. *Let $\varphi \in K$ and $A_\varphi = \langle AP, Q, \mathcal{D}, \delta, \varphi\rangle$, we define A^D_φ from A_φ as follows*

$$A^D_\varphi = \langle AP, Q, \mathcal{D}, \delta^D, \varphi\rangle$$

With for all $\Box\varphi_1 \in Q^\Box$,$\delta^D(\Box\varphi_1, 0) = (\emptyset, \bot, \bot)$ and $\delta^D(q,k) = \delta(q,k)$ for all $(q,k) \neq (\Box\varphi_1, 0)$

Proposition 8. *$L(A^D_\varphi) \neq \emptyset$ iff there is an open branch in $\mathcal{T}^D_\varphi$.*

Proof. The proof follows the same reasoning that the proof for the Proposition 7. □

The object A_φ is central in this work. We will reuse it in further section for the cases KT and KD.

6.3 Reducing KT-Sat to KA-emptiness

In this section, we prove the following result.

Theorem 7. *Given a K-formula φ we can construct in linear running time a K-automata A^T_φ such that φ is KT-satisfiable if, and only if, $L(A^T_\varphi) \neq \emptyset$.*

Let φ be a K-formula. Each subformula of φ will give rise to a state of automaton A^T_φ, and the transition function δ is inspired from the tableau rules for KT (Section 3). More formally,

Definition 13. *Let $\varphi \in K$ and $A_\varphi = \langle AP, Q, \mathcal{D}, \delta, \varphi\rangle$, we define A^T_φ from A_φ as follows*

$$A^T_\varphi = \langle AP, Q, \mathcal{D}, \delta^T, \varphi\rangle$$

with for all $\Box\varphi_1 \in Q^\Box$ and for all $k \geq 1$,$\delta^T(\Box\varphi_1, k) = (\emptyset, \bigwedge_{i=0}^{k-1}(i, (\varphi_1)) \wedge \bigwedge_{i=0}^{k-1}(\varepsilon, (\varphi_1)), \bot)$ and $\delta^D(q,k) = \delta(q,k)$ for all $(q,k) \neq (\Box\varphi_1, n)$

Proposition 9. *$L(A^T_\varphi) \neq \emptyset$ iff there is an open branch in $\mathcal{T}^T_\varphi$.*

Proof. The proof follows the same reasoning that the proof for the Proposition 7. □

6.4 Reducing K4-Sat to KA-emptiness

In this section, we prove the following result.

Theorem 8. *Given a K-formula φ we can construct in linear running time a SL-automaton A^4_φ such that φ is K4-satisfiable if, and only if, $L(A^4_\varphi) \neq \emptyset$.*

Let φ be a K-formula. Each subformula of φ will give rise to a state of automaton A^4_φ, and the transition function δ is inspired from the tableau rules for K4 (Section 3). More formally,

Definition 14. *for $\varphi \in K : A^4_\varphi = \langle AP, Q, \mathcal{D}, \delta, \varphi \rangle$ is defined by:*

- $AP =$ *the set of propositions that appear in* φ
- $Q = Closure(\varphi)$ *and*
  - $Q_\Box = \{\Box\psi | \Box\psi \in Closure(\varphi)\}$
  - $Q_\Diamond = \{\Diamond\psi | \Diamond\psi \in Closure(\varphi)\}$
  - $Q_b = Q \setminus (Q_\Box \cup Q_\Diamond)$
- *with* $\succ$ *as* "is a subformula of"
- $\mathcal{D} = \{0, ..., |Q_\Diamond|\}$
- φ *is the initial state*
- δ*:*
  - $\forall k \in \{1, ..., |Q_\Diamond|\}$*:*
    - $\forall p \in AP, \delta(p, k) = (\{p\}, \top, \bot)$
    - $\forall(\neg\psi) \in AP, \delta(\neg\psi, k) = (\{p\}, \bot, \top)$
    - $\delta(\varphi_1 \wedge \varphi_2, k) = (\emptyset, (\varepsilon, \varphi_1) \wedge (\varepsilon, \varphi_2), \bot)$
    - $\delta(\varphi_1 \vee \varphi_2, k) = (\emptyset, (\varepsilon, \varphi_1) \vee (\varepsilon, \varphi_2), \bot)$
    - $\delta(\Diamond\varphi_1, k) = (\emptyset, \bigvee_{i=0}^{k-1}(i, \varphi_1), \bot)$
    - $\delta(\Box\varphi_1, k) = (\emptyset, \bigwedge_{i=0}^{k-1}((i, \varphi_1) \wedge (i, \Box\varphi_1)), \bot)$
  - $k = 0$*:*
    - $\forall p \in AP, \delta(p, 0) = (\{p\}, \top, \bot)$
    - $\forall p \in AP, \delta(\neg p, 0) = (\{p\}, \bot, \top)$
    - $\delta(\varphi_1 \wedge \varphi_2, 0) = (\emptyset, (\varepsilon, \varphi_1) \wedge (\varepsilon, \varphi_2), \bot)$
    - $\delta(\varphi_1 \vee \varphi_2, 0) = (\emptyset, (\varepsilon, \varphi_1) \vee (\varepsilon, \varphi_2), \bot)$

* $\delta(\Diamond\varphi_1, 0) = (\emptyset, \bot, \bot)$
* $\delta(\Box\varphi_1, 0) = (\emptyset, \top, \bot)$

Proposition 10. *$L(A^4_\varphi) \neq \emptyset$ iff there is an open branch in $\mathcal{T}^4_\varphi$.*

Proof. the proof follows the same reasoning that the proof for the Proposition 7. Except that we have to deal with the correspondence of infinite branch with infinite run. □

7 Conclusion

We have shown an automata-theoretic approach to address and solve satisfiability problems of axiomatic extensions of modal logic K. To do so we have designed subclasses of alternating automata which, to our knowledge, have not been considered in the literature so far. The PSPACE-completeness of the non-emptiness problems for these automata essentially demonstrates their optimality. Also, we have established a tight correspondence between runs of automata and branches of the standard tableau construction for the logics; we believe this correspondence highlights the relationship between the two formalisms.

It is legitimate to wonder whether the proposed classes of automata fully characterize the considered logics in the sense that any automata in the class is equivalent to some automaton A_φ for some formula φ. This question must be investigated in a further work.

We are confident our approach extends to multimodal logics obtained by fusioning standard modals logics (for example the fusion of K and KT). It would be interesting to investigate combinations of modal logics with interactions (such as requiring that $\Box_1\Box_2\varphi \Rightarrow \Box_2\Box_1\varphi$ holds), although tableau methods have already shown their limitations. Also, it should be possible to exploit the tableau method for public announcement logic [7] for designing a subclass of automata that does the job.

Actually, philosophers and logicians also have some logics that use fixpoints, in particular the operator *Common Knowledge* in multi-agent systems as $S5_n$ [9]. In this context, an automata-based method would be quite elegant and may lead to interesting results.

It is well known that alternating tree automata easily deal with arbitrary fix-points (using the powerful parity acceptance) [5]. A natural try would be to extend K-automata and the SL-automata in order to deal with parity acceptance conditions. This is what we plan to work on in the near future.

References

[1] Beth and Evert W. *Semantic entailment and formal derivability.* Oxford University Press, 1955.

[2] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic.* Cambridge University Press, Cambridge, 2001.

[3] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.

[4] D.M. Gabbay, R. Hähnle, and J. Posegga. *Handbook of Tableau Methods.* Springer, 1999.

[5] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500. Springer, 2002.

[6] Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000.

[7] Jan Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[8] Wolfgang Thomas. *Languages, Automata, and Logic.* Springer, 1996.

[9] Wiebe van der Hoek and Michael Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.