Proofs for temporal logics: focus on the multi-agent case

Serenella Cerrito

Proofs for temporal logics: focus on the multi-agent case

Serenella Cerrito

Université Paris-Saclay, Université D'Evry, IBISC, 91025, Evry, France
serenella.cerrito@univ-evry.fr

Abstract. In this work we present and discuss some features of proofs in the case of temporal logics. More precisely, we discuss the relation between proofs for temporal logics built in two distinct frameworks: sequent calculi and tableau based calculi, taking as a typical example of temporal logic propositional LTL (Linear Temporal Logic). This work is a survey focussing on the comparison between sequent calculi and tableaux proofs. We also illustrate with some details the tableau based proofs for ATL* (the most expressive logic of the family of Alternating-Time Temporal Logics) proposed by Amélie David in 2015. Such a logic takes into account the presence of several agents that can cooperate (or not) in a given scenario to achieve some temporal objectives. Up to our knowledge, no proof-system in the sequent style is known for ATL*.

1 Introduction

1.1 Philosophical context

The study of the notion of "proof" and of the related question of what makes an argument valid is central in gnoseology and has far roots. Aristotle devoted a great amount of work to study the nature of valid argumentations, and interest for these problems continued to be alive during Middle Age and Renaissance. But certainly it acquired a new importance towards the end of 19th century and blossomed in the 20th century, thanks also to the fundamental work of Gottlob Frege [Fre79], the father of formal logic, which allowed for the view of a mathematical proof as a formal object. In 1922 David Hilbert writes:
"*we must make the concept of specific mathematical proof itself object of investigation, just as also the astronomer pays attention to his place of observation, the physicist must care about the theory of his instrument, and the philosopher criticizes reason itself*" [Hil22]

With Hilbert proof theory was born. However, a formalization of a logical proof for Hilbert is a linear succession of formulae leading from assumptions to the conclusion to be proved (and this view actually comes from Frege's work). Gerhard Gentzen [Sza71], in the early 1930s, aims to formulate proofs so as "to study the structure of mathematical proofs as they appear in *practice*[1]".

[1] The emphasis is ours.

Moreover, he presents proofs as trees, that allows for a form of modularity: subtrees can be combined in new ways to generate new proofs. The calculi defined by Gentzen are *sequent* calculi and *natural deduction* calculi; in both cases he provides combinatorial transformations of any proof into a given direct "canonical form" (syntactic cut elimination for sequent calculi and normalization for natural deduction). As observed in [NvP14]:

Gentzen's analysis of the structure of proofs in logic was a perfect success. He was able to show that the means of proving a logical theorem can be restricted to those that concern just the logical operations that appear in the theorem. Instead of logical axioms, there are just rules of inference, separately for each logical operation such as conjunction or implication, to the said effect. Logic in the whole is seen as method for moving from given assumptions to a conclusion. The Fregean tradition, instead, presented logic as consisting of a basic stock of logical truths, namely the axioms of logic, together with two rules by which new logical truths can be proved from the axioms.

Gentzen's approach allows for a deep study on the nature of proofs, giving precise grounds, for instance, to meaningfully ask the question : 'In which case a proof of a given property is simpler than another proof of the same result?" Thus with Gentzen's work proof theory made a very important progress, and new questions were opened. In Hilbert's program the role of proof theory was essentially to prove the consistency of mathematics, and it is well known that Gödel's incompleteness theorem destroyed Hilbert's dream of using proof theory to provide a finitist foundation of mathematics. In [Pra71] D. Prawitz proposed the name of *general proof theory* as an appropriate name for a field where proofs are studied in their own right, in contrast to *reductive proof theory* where the study is a tool for reductive aims. It was Gentzen that made general proof theory really possible.

Nowadays we know a great number of different styles of formulation of formal systems aiming at proving theorems, beside Hilbert style axiomatic systems, namely sequent calculi and natural deduction. For instance, resolution systems (founded on [Rob65]), tableaux (see [DGHP99]), connection methods ([OB03]), proof-nets (see [Gir87]), etc. Tableaux systems have a semantical origin, since they are rooted in the work on semantics by Evert William Beth's [Bet56,Bet59]; the name "tableaux" for these calculi comes from Beth himself. But, despite the fact that sequent calculi and tableaux calculi are born in different contexts, respectively syntactical and semantical, the two approaches began to meet in the late sixties: it was realized that classical semantic tableau systems and classical Gentzen systems were essentially the same thing. This is observed both by Melvin Fitting and Rajeev Gore in their chapters in [DGHP99]. Indeed, tableau calculi have a formulation that is not as appropriate as sequent calculi to study the nature of proofs (both from a mathematical and a philosophical point of view) but that is particularly apt to do mechanical proof search.

In the case of modal logics, however, this strong link between sequent and tableau proofs holds for many logics of the family but not for all of them. It does not hold for some modal logics that are quite used in computer science

to model computer systems and their dynamic behavior. This is typically the case of temporal logics, as this work emphasizes. General proof theory is quite alive nowadays, see, for an instance, [StL19]; however proof-theorists have not devoted a lot of attention to temporal proofs, in general.

1.2 Aim and structure of this work

In this work we focus on temporal proofs.

Very generally speaking, a modality is an expression like "necessarily" (often noted $\Box$) or "possibly" (often noted $\Diamond$) and modal logics study the deductive behavior of expressions as "it is necessary that" and "it is possible that". Among the most well known modal logics there are the propositional logics K, $\mathsf{K4}$, T and $\mathsf{S4}$. However a lot of different logics, containing many different modal operators, belong nowadays to the very broad family of modal logics, and include, for instance, logics for belief, deontic logics, dynamic logic and temporal logics.

Among modal logics, temporal logics, aiming to model time and to reason on it, have acquired a lot of importance in computer science. This is quite natural because an execution of a computer system, or of a program, performs actions in time and one may be interested to express and check properties concerning a temporal ordering between these actions. But time can receive several different abstract representations. For instance it can be taken to be continuous or discrete, one can suppose that there is an initial instant (corresponding to a starting point of a computation) or not, that the future is potentially infinite or not, that each state of a computation has exactly one next state (this is the case of LTL, described later on) or that the time is branching, so that "several futurs are possible" (see CTL or ATL, also described below). Temporal logics are a typical example of modal logics were the notion of proof is, in general, not completely understood, and not much studied by proof theorists.

The general aim of this work is to make some remarks on proofs for modal logics formulated in two styles, namely by sequents and by tableaux, so to highlight their differences and their similarities. We emphasize that for "standard" modal logics like K, $\mathsf{K4}$, T and $\mathsf{S4}$, actually tableaux are just cut-free sequent proofs under a given formulation of sequent rules, while this link between tableaux and sequent proofs becomes problematic, as a matter of fact, when temporal logics, typically LTL, are considered. We investigate the intrinsic reasons -if any- for this gap, and we shortly describe some works aiming to overcome it. Then, we describe with some details a tableau calculus for the alternating-time temporal logic ATL^* (a multi-agent temporal logic) that has been firstly published in [Dav15a], building on [CDG14,CDG15,GvD06]; up to our knowledge, there exists no sequent calculus deciding ATL^*.

In Section 2 we recall the syntax and the semantics of some modal logics, namely K, $\mathsf{K4}$, T, $\mathsf{S4}$, LTL and, briefly, CTL. In Section 3 we illustrate the very strong connection existing between tableaux proofs and sequent proofs for $\mathsf{S4}$. In Section 4 we present a very well known tableau system for LTL that works in two phases, thereby being very far from the philosophy of usual sequent calculi.

We mention, however, some more recent work done to get one-pass tableaux for LTL and/or sequent calculi for this logic. We describe the essential features of the two-pass tableau calculus proposed by Amélie David in [Dav15a] for the logic ATL* in Section 6. We conclude by pointing at some lines of future research.

2 Preliminaries on modal and temporal logics

2.1 Modal logics K, K4, T and S4

In this section we recall the basic syntactical and semantical notions for some well known propositional modal logics: K, K4, T and S4, although in the sequel we will focus on S4 only.

Unless explicitly differently stated, here and through the whole article we will use lower-case latin letters for propositional variables and lowercase Greek letters for formulae.

Let P be a set of propositional variables. The set of formulae of a basic modal (propositional) language can be defined via the following grammar:

$$\varphi := p \mid \top \mid \bot \mid (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid (\Box\varphi) \mid (\Diamond\varphi)$$

where $p \in P$.

The set $P \cup \{\top, \bot\}$ is the set of *atomic formulae*. When there is no risk of ambiguity we wil omit parentheses. As usual, $\varphi \to \psi$ can be defined as $\neg\varphi \vee \psi$

Formulae are interpreted on *interpreted transition systems* –called also *Kripke interpretations*[2] (from [Kri63b])– defined as follows.

Definition 1. *A frame $\mathcal{F}$ is a pair $(W, \mathcal{R})$ where W is a non-empty set worlds and $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ (transition relation).*

An interpreted transition system (ITS) $\mathcal{M}$ is a pair $(\mathcal{F}, \mathsf{L})$ where $\mathsf{L} : W \to \mathcal{P}(P)$ is a world description function defining for every world in W the set of atomic propositions true at that world.

The set W is called the domain of the ITS. For $w, w' \in W$, whenever $w\mathcal{R}w'$ we say that w' is accessible from w.

Let w be a world in an ITS $\mathcal{M}$. The notion of a formula φ being satisfied (or true) at w, noted $\mathcal{M}, w \models \varphi$ is inductively defined. We omit here the obvious boolean cases (we just recall that $\top$ is the constant for truth and $\bot$ the constant for falsity) and explicit just the modal ones:

Definition 2.

- $\mathcal{M}, w \models \Box\varphi$ *if and only if for all w' such that $w\mathcal{R}w'$ we have $\mathcal{M}, w' \models \varphi$,*
- $\mathcal{M}, w \models \Diamond\varphi$ *if and only if for some w' such that $w\mathcal{R}w'$ we have $\mathcal{M}, w' \models \varphi$.*

[2] The first name is mostly used in computer science, while the second mostly in mathematics and philosophy.

A formula φ is *satisfiable* if there are an ITS $\mathcal{M}$ and a world w of $\mathcal{M}$ such that $\mathcal{M}, w \models \varphi$, otherwise it is said to be *unsatisfiable*. It is *valid* if and only if $\neg\varphi$ is unsatisfiable. Two formulae φ and ψ are *logically equivalent*, noted $\varphi \equiv \psi$, when for any ITS $\mathcal{M}$ and any world w we have $\mathcal{M}, w \models \varphi$ if and only if $\mathcal{M}, w \models \psi$.

Clearly for any φ we have $\Box\varphi \equiv \neg\Diamond\neg\varphi$ and $\Diamond\varphi \equiv \neg\Box\neg\varphi$. Hence modal formulae can be put in negation normal form (*fnn*), where negation applies only to atomic formulae.

When no hypothesis is made on frames, then the valid formulae are the theorems of the logic K, the smallest normal modal logic. By taking only $\Box$ as primitive modal operator, and $\Diamond$ as a defined one, the logic K can be axiomatically characterized by the addition of the K axiom schema:

$$\Box(\varphi \to \psi) \to ((\Box\varphi) \to (\Box\psi))$$

and of the *Generalization* rule schema:

$$\frac{\varphi}{\Box\varphi}\ (GEN)$$

to an Hilbert style axiomatization of classical propositional logic.

When the transition relation is supposed to be reflexive, which can be expressed by the addition of the axiom schema T:

$$(\Box\varphi) \to \varphi$$

one obtains the logic T.

Frames corresponding to the logic $\mathsf{K4}$ are such that the only hypothesis is that the transition relation is transitive, which is expressed by the axiom schema 4:

$$(\Box\varphi) \to (\Box\Box\varphi)$$

Finally, the logic $\mathsf{S4}$ is such that transition relation is supposed to be both transitive and reflexive; therefore it can be axiomatically characterized by the addition of both the axioms T and 4 to an Hilbert style axiom systems of classical propositional logic. The logic $\mathsf{S4}$ is the one of the above considered modal logics that is the most suitable to model time: $\Box\varphi$ can be read as: φ holds always in the future (including the present time). However $\mathsf{S4}$ frames can contain cycles, which is counter-intuitive if time has to be modelled; this defect is absent in LTL, considered in the next section.

The notion of frame can be generalized so to be a pair $(W, \{R_a \mid a \in A\})$, where A is non-empty set of labels and each $R_a \subseteq W \times W$ is a distinguished transition relation, allowing to interpret a specific set of modal operators. Then one obtains *multi-modal logics*. The logics K, $\mathsf{K4}$, T and $\mathsf{S4}$ are then particular cases of multi-modal logics corresponding to frames equipped of just one transition relation, interpreting the modal operators $\Box$ and $\Diamond$.

2.2 Temporal Logics

A variety of temporal logics exists, widely used in computer science to model computer systems or programs and to express their properties. Many of them are variations and/or extensions of the propositional logic LTL (Linear Temporal Logic) first proposed by Amir Pnueli in [Pnu77] and known also under the name PTL.

LTL. Let P be a set of propositional variables. The set of formulae of a LTL (propositional) language can be defined via the following grammar:

$$\varphi := a \mid \top \mid \bot \mid (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid (\Box\varphi) \mid (\Diamond F) \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

where a is an atomic formula.

Definition 3. *A state is a subset of P. An interpretation $\mathcal{M}$ of* LTL *is a surjective function $\mathbb{N} \to 2^P$ enumerating states, possibly with repetitions (necessarily when P is finite): $s_0, s_1, s_2, \ldots$. The state s_0 is called initial state.*

Let us note $\mathcal{M}_i$ the element a of such a sequence at position i. Then the satisfaction relation is inductively defined as follows (omitting the inductive boolean cases):

- If φ is atomic, $\mathcal{M}_i \models \varphi$ if and only if either φ is $\top$ or φ is a propositional variable p and $p \in \mathcal{M}(i)$.
- $\mathcal{M}_i \models \bigcirc\varphi$ if and only if $\mathcal{M}_{i+1} \models \varphi$.
- $\mathcal{M}_i \models \Box\varphi$ if and only if $\mathcal{M}_j \models \varphi$ for each $j \geq i$.
- $\mathcal{M}_i \models \Diamond\varphi$ if and only if there is a $j \geq i$ such that $\mathcal{M}_j \models \varphi$.
- $\mathcal{M}_i \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a $j \geq i$ such that $\mathcal{M}_j \models \varphi_2$ and, for any n where $1 \leq n < j$, $\mathcal{M}_n \models \varphi_1$.

Observe that LTL can be seen as a bimodal logic: the domain W can be taken to be the set of elements of any ω-sequence of states, L is built in, since the propositions true at the world $\mathcal{M}_i$ are exactly the elements of $\mathcal{M}_i$ (that is a state), and we have two transition relations $\mathcal{R}_{\geq}$ and $\mathcal{R}_{Next}$, where $\mathcal{M}_i \, \mathcal{R}_{\geq} \, \mathcal{M}_j$ iff and only if $i \leq j$ and $\mathcal{M}_i \, \mathcal{R}_{Next} \, \mathcal{M}_j$ iff and only if $j = i+1$. The accessibility relation $\mathcal{R}_{\geq}$ allows for the interpretation of the modal operators $\Box, \Diamond$ and U (until), while $\mathcal{R}_{Next}$ allows for the interpretation of the other specific operator, $\bigcirc$ (Next). The relation $\mathcal{R}_{\geq}$ is reflexive and transitive (as the transition relation of S4), while $\mathcal{R}_{Next}$ is functional: given any state s there exists exactly one s' such that $s \, \mathcal{R}_{Next} \, s'$. Actually $\mathcal{R}_{\geq}$ is the reflexive and transitive closure of $\mathcal{R}_{Next}$ and an LTL frame is isomorphic to $(\mathbb{N}, \leq, Succ)$, where, for $n, m \in \mathbb{N}$, $n \; Succ \; m$ if and only if $m = n + 1$. There is exactly one such a frame.

Thus, modulo the use of "LTL-interpretation" at the place of "ITS", the notions of satisfiability, validity and logical equivalence for LTL keep the same as in Section 2.1[3].

[3] Alternatively, satisfiability can be defined as 'truth at the initial state of some interpretation", but a formula is satisfiable according to the first definition if and only if it is so according to the second one

Let us observe that $\neg \bigcirc \varphi \equiv \bigcirc \neg \varphi$, $\Diamond \varphi \equiv \top \mathsf{U} \varphi$ and $\Box \varphi \equiv \neg(\top \mathsf{U} \neg \varphi)$. The temporal operator R (*release*), not given as primitive in the above grammar, can be defined by: $\varphi \mathsf{R} \psi \ = \ (\Box \psi) \vee (\psi \mathsf{U} (\psi \wedge \varphi))$ and it easy to show that $\neg(\varphi \mathsf{U} \psi) \equiv (\neg \varphi) \mathsf{R} (\neg \psi)$. This allows for rewriting any LTL formula as an equivalent formula in fnn.

Among the remarkable equivalences we find the *fixed point equivalences*:
$\Box \varphi \ \equiv \ \varphi \wedge \bigcirc \Box \varphi$
$\Diamond \varphi \ \equiv \ \varphi \vee \bigcirc \Diamond \varphi$
$\varphi \mathsf{U} \psi \ \equiv \ \psi \vee (\varphi \wedge \bigcirc (\varphi \mathsf{U} \psi))$.

We underline that all the instances of the induction schema:

$$(IND): \quad (p \wedge \Box(p \rightarrow \bigcirc p)) \rightarrow \Box p$$

are valid.

It is worthwhile observing that the fixed point equivalences hold because the accessibility relation $\mathcal{R}_{\geq}$, interpreting $\Box$, is the reflexive and transitive closure $\mathcal{R}_{Next}$, interpreting $\bigcirc$. Both the fixed point equivalences and the induction schema have a recursive nature, given to the fact that the LTL frame is isomorphic to $(\mathbb{N}, \leq, Succ)$, as we noted before. This feature of LTL semantics constitutes a difficulty for the definition of satisfactory sequent calculi, as we will discuss in the sequel.

Branching Temporal Logics In the case of LTL time is modeled as being linear, and any numbered state s_i has exactly one successor. This is not the case for the *Branching Temporal Logics*, where at any instant of time there are several "possible futures". We shortly describe here the most powerful of these logic, CTL^* (*Computation Tree Logic*, [CES86]).

The definition of the grammar of formulae involves a mutual recursion between *state formulae*, that we will be noted by lower case Greek letters, and *path formulae*, noted by upper case Greek letters.

State formulae: $\psi := a \ \mid \ (\psi \wedge \psi) \ \mid \ (\psi \vee \psi) \ \mid \ (E\Phi) \ \mid \ (A\Phi)$
where a is an atomic formula

Path formulae: $\Phi := \psi \ \mid \ (\neg \Phi) \ \mid \ (\Phi \wedge \Phi) \ \mid \ (\bigcirc \Phi) \ \mid \ (\Box \Phi) \ \mid \ (\Phi \mathsf{U} \Phi)$

This syntactic distinction between two classes of formulae reflects semantics, where time is modeled so that several different infinite paths, describing possible future executions of a system or program, stem from a given state at a given instant of time. The operators E and A are quantifiers over paths, respectively existential and universal. A state formula, for instance $E\Box\Diamond p$ – stating the existence of least a path where p holds infinitely often –, is evaluated at a state, the (possible) origin of the claimed path. Path-formulae, as for instance $\Box\Diamond p$, on the contrary, state a property of the path on which they are evaluated. It is worthwhile observing that any state formula is also a path formula, while the converse is false. Let us observe that in LTL, given the uniqueness of the path corresponding to the ω-sequence of states of a given interpretation, this kind of

distinction would be meaningless. By default, "CTL* formula", *tout court*, means "state formula".

We do not give here the formal semantics of CTL*, that can be found in [DGL16], because our discussion of temporal sequent calculi and tableaux will focus on LTL and also because in a forthcoming section (Section 6) we describe with some details the logic ATL* that generalizes CTL* to the multi-agent case, so that CTL* semantics is subsumed by ATL* semantics.

Among the distinguished fragments of CTL* one finds CTL, which obeys the constraint that any path quantifier is followed by a temporal operator and, *vice-versa*, any temporal operator is preceded by a path quantifier. This logic is largely used to model computer systems because of its good computational properties.

Finally, it is is worthwhile observing that, in the philosophical context, there are several possible semantics for branching-time logics: see [Bel01].

3 Sequent Proofs and Tableaux for S4

In this section we recall sequent calculi and tableaux calculi allowing one to prove modal formulae of the logic S4, with the aim to illustrate the deep links existing between these two proof styles. It is worthwhile reminding that similar links exist also between sequent calculi and tableaux calculi for a variety of modal logics, including K, T and K4.

Sequent calculi for classical and intuitionistic first order logics are rooted in the fundamental work on proof theory for classical and intuitionistic logic by Gerhart Gentzenin of the thirties of the 20th century by [Sza71]. The first attempt to formulate sequent calculi for some modal logics can be found in [Cur52] and [OM57]. Tableaux systems have a semantical origin, since they originate from [Bet56,Bet59], although their development for modal logics had to wait for the definition of a proper modal semantics by S. Kripke [Kri63a]. Despite the fact that sequent calculi and tableaux calculi are born in different contexts, respectively syntactical and semantical, the two approaches began to meet in the late sixties when it was realized that classical semantic tableau systems and classical Gentzen systems were essentially the same thing (see the chapters by Melvin Fitting and Rajeev Gore in [DGHP99]).

For the sake of concision we do not formulate here sequent or tableau rules for $\top$ and $\bot$.

3.1 Two sided Sequent Calculi for S4

Since Gentzen's original formulation, sequent calculi have been formulated in many different manners. Let us summarize the main essential choices:

1. The main datatype structure can be a sequence, a multi-set or a set of formulae;

2. The structural rules *Exchange*, *Weakening* and *Contraction* can be either explicit rules or else can be embedded in the logical rules and the data structures;
3. Following a terminology introduced by J-Y. Girard [Gir87], each of the rules for the binary connectives $\wedge$ and $\vee$ can have either an *additive* or a *multiplicative* formulation;
4. The cut rule can be present (and its elimination is possibly demonstrated) or not, and in the second case the calculus is by definition cut-free;
5. Formulae are either "pure" – just expressions of the grammar of the objet language – or accompanied by labels, encoding semantical informations (*labelled sequent calculi*);
6. Traditionally calculi are "two-sided" (a sequent has a left part and a right part), but also one sided-versions are possible.

In our presentation we choose:

- to use sets of pure formulae as data-structures;
- to embed the structural rules in the calculus (when needed for completeness);
- to give an hybrid additive-multiplicative formulation maximizing determinism when doing proof-search;
- finally, to directly formulate cut-free calculi.

Our choices are motivated by the desire of making particularly evident the connection between sequent calculi and tableaux for S4; an analogous link holds between sequent calculi and tableaux for K, K4 and T. That is, we aim at showing that for these logics tableaux are essentially a version of sequent calculi that is well suited for automated proof search although not very appropriate for proof theory.

We present both a two-sided version of a sequent calculus and a one-sided one.

Some comments must be made on our choice of sets as data structures. In Gentzen's original work a (two-sided) sequent is a couple of sequences of formulae and the structural rules govern the general use of such formulae. Roughly, *Exchange* states that the order of formulae does not matter, so it amounts to use multi-sets rather than sequences as data structures. In the sequel we always implicitly assume Exchange. The *Contraction* rules allow for the re-use of formulae when building a proof starting from the conclusion sequent and moving up toward the leafs (the axioms), that is, to duplicate the principal formula of a rule as soon as the rule is used, while *Weakening* rules allow for "forgetting" a formula while doing such a proof search. That is, Contraction and Weakening rule are (Γ and Δ being sequences of formulae):

$\dfrac{\Gamma, \varphi, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ (Contraction\ L)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}\ (Contraction\ R)$
$\dfrac{\Gamma \Rightarrow \Delta}{\Gamma, \varphi \Rightarrow \Delta}\ (Weakening\ L)$	$\dfrac{\Gamma, \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}\ (Weakening\ R)$

If, differently from Gentzen's original formulation, one takes a sequent to be a couple of sets (rather than sequences or multisets) of formulae, then, *de facto*, Contraction rules donot make sense any longer, since any formula can appear at most once in a set. In this case rule formulation becomes context-dependent (see the discussion in [Nv16]). For instance, a precise and explicit formulation for the left rule for conjunction ($\wedge L$) should be:

$\dfrac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \quad if \ \ \varphi \wedge \psi \in \Gamma$
$\dfrac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta} \ \ if \ \ \varphi \wedge \psi \notin \Gamma$

This is rather cumbersome. Moreover, as a consequence, syntactical proofs of meta-theorems (as cut-elimination) become technically involved, which is certainly a backdraw for proof-theory. Here, we do take sets as data structures but we set the convention that Γ, φ means set-union : $\Gamma \cup \{\varphi\}$; this allows us to avoid the context-depended formulation, that splits each logical rule in two cases, mentioned above. Once again, we remind the reader that our technical choice is motived only by the desire of making very explicit the link between tableaux and sequent calculi, not by proof-theoretical concerns. However, it is important to observe that even with these conventions when we want to allow for the re-use of a formula φ when building a proof starting from the conclusion sequent and moving up, we must explicitly duplicate φ, possibly embedding this duplication in a logical rule defining a logical operator $\mathcal{O}$ (see later, in Figure 3.1, our left rule for $\Box$ and our right rule for $\Diamond$).

After these preliminary remarks we are ready to present more in details the version of sequent calculi chosen here.

We will use Greek capital letters to denote finite sets of formulae[4]. A (two-sided) *sequent* is a couple of -possibly empty- sets of formulae Γ, Δ written $\Gamma \Rightarrow \Delta$; Γ is called *antecedent* of the sequent and Δ is called *consequent*. Braces are usually omitted and Γ, φ denotes $\Gamma \cup \{\varphi\}$.

[4] This notation is used also for temporal path formulae, later on in this work, but the meaning of any given occurrence of a Greek capital letter will be clear from the context.

The intuitive interpretation of a sequent $\varphi_1, ..., \varphi_n \Rightarrow \psi_1, ..., \psi_m$, $n, m \geq 0$, is that under the hypotheses φ_1 and ... φ_n one can get at least one of the conclusions $\psi_1, ..., \psi_m$. Let us note Γ_φ the formula $\varphi_1 \wedge ... \wedge \varphi_n$ ($\top$ in the case $n = 0$) associated to an antecedent $\Gamma = \varphi_1, ..., \varphi_n$, and Δ_ψ the formula $\psi_1 \vee ... \vee \psi_m$ ($\bot$ in the case $m = 0$) associated to a consequent $\Delta = \psi_1, ..., \psi_m$; the formula associated to the sequent $\Gamma \Rightarrow \Delta$ is $\Gamma_\varphi \rightarrow \Delta_\psi$. A sequent is valid if and only if its associated formula is valid. In particular, a sequent $\Rightarrow \psi$ is valid if and only if the formula ψ is valid and a sequent $\psi \Rightarrow$ is valid if an only if ψ is unsatisfiable.

Sequent calculi are mainly thought as *direct* proof-systems of valid formulae: given a set of inferences rules, including the axioms (rules with zero premisses), a proof of (the validity of) a sequent $\Gamma \Rightarrow \Delta$ is a finite tree such that: each node is a sequent, the root is the sequent $\Gamma \Rightarrow \Delta$, the leaves are axioms and the children of a sequent $\mathcal{S}$ are the sequents $\mathcal{S}_1, ..., \mathcal{S}_n$ such that $\frac{\mathcal{S}_1, ..., \mathcal{S}_n}{\mathcal{S}}$, $n \geq 1$ is an inference rule. Such trees are usually written with the root down and the leaves up.

In the two-sided formulation of sequents each logical operator $\mathcal{O}$ is equipped with a left rule $(\mathcal{O}, L)$ and a right rule $(\mathcal{O}, R)$. Each rule can be read either top-down, as an inference rule allowing for the deduction of the sequent under the line (the conclusion) from the sequent (the sequents) upside – the premise or premises –, or else down up, as a proof-search rule, searching for a proof of a given sequent $\mathcal{S}$ starting from it and going upwards toward axioms. If a rule is seen as a proof search rule, it enables to replace the current goal to be proved –the lowest sequent– with with the new goal(s) to be proved – the uppermost sequent(s)–.

The axiom schemas and the propositional rules schemas are given in Figure 1. This formulation of the rules is the same as in the sequent calculus called G in [Gal15] that was originally formulated by S. C. Kleene. Readers familiar with linear logic terminology will recognize that $(\wedge L)$ and $(\vee R)$ have a multiplicative formulation while $(\wedge R)$ and $(\vee L)$ have an additive formulation. This formulation of the rules allows for a deterministic proof search. Indeed, all the rules in the figure are not only sound but *reversible*: the sequent conclusion is valid (provable) if and only if the premises are valid (provable).

The rules for the modal operators are given in Figure 2 where the notations $\Box\Gamma$ and $\Diamond\Gamma$ mean, respectively, the set of formulae $\{\Box\varphi \mid \varphi \in \Gamma\}$ and $\{\Diamond\varphi \mid \varphi \in \Gamma\}$.

It is worthwhile observing that $(\Box L)$ and its dual $(\Diamond R)$ embed contraction, while $(\Box R)$ and its dual $(\Diamond L)$ embed (several applications of) Weakening: this is necessary for completeness. Observe also that the rules $\Box R$ and $\Diamond L$ are not reversible, while all the other rules are so.

As an example of proof in the calculus we give here a proof of the validity of the formula $\varphi = (\Box p) \rightarrow \Box\Box p$, that is an instance of the axiom schema 4, in a refutation style. That is, we give a proof of the sequent $(\Box p) \wedge \Diamond\Diamond\neg p \Rightarrow$, where the formula $(\Box p) \wedge \Diamond\Diamond\neg p$ is equivalent to $\neg\varphi$ and the formula associated to the sequent is therefore equivalent to $\neg\varphi \rightarrow \bot$. Although the natural philosophy of sequent calculi is the construction of direct proofs, they can be used also

$\dfrac{}{\Gamma, \varphi \Rightarrow \Delta, \varphi}\ (Axioms)$	
$\dfrac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma, \neg\varphi \Rightarrow \Delta}\ (\neg L)$	$\dfrac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi}\ (\neg R)$
$\dfrac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta}\ (\wedge L)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}\ (\wedge R)$
$\dfrac{\Gamma, \varphi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta}\ (\vee L)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi}\ (\vee R)$

Fig. 1. Sequent Axioms and Sequent Propositional Rules

$\dfrac{\Gamma, \varphi, \Box\varphi \vdash \Delta}{\Gamma, \Box\varphi \vdash \Delta}\ (\Box L)$	$\dfrac{\Box\Gamma \Rightarrow \Diamond\Delta, \varphi}{\Lambda, \Box\Gamma \Rightarrow \Lambda', \Diamond\Delta, \Box\varphi}\ (\Box R)$	$\dfrac{\Box\Gamma, \varphi \Rightarrow \Diamond\Delta}{\Lambda, \Box\Gamma, \Diamond\varphi \Rightarrow \Lambda', \Diamond\Delta}\ (\Diamond L)$	$\dfrac{\Gamma \Rightarrow \Delta, \Diamond\varphi, \varphi}{\Gamma \Rightarrow \Delta, \Diamond\varphi}\ (\Diamond R)$

Fig. 2. S4: Sequent Rules for $\Box$ and $\Diamond$

to construct proofs by refutation. We choose here to prove ϕ by refuting $\neg\phi$ because this example will help to see that tableau proofs for S4, that are always refutations, can be seen as sequent proofs.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Box p, p \Rightarrow p}\ (axiom)}{\Box p, p, \neg p \Rightarrow}\ (\neg L)}{\Box p, \neg p \Rightarrow}\ (\Box L)}{\Box p, p, \Diamond\neg p \Rightarrow}\ (\Diamond L)}{\Box p, \Diamond\neg p \Rightarrow}\ (\Box L)}{\Box p, p, \Diamond\Diamond\neg p \Rightarrow}\ (\Diamond L)}{\Box p, \Diamond\Diamond\neg p \Rightarrow}\ (\Box L)}{(\Box p) \wedge (\Diamond\Diamond\neg p) \Rightarrow}\ (\wedge L)$$

3.2 Tableau calculi for S4

Also tableaux come in many different formats. Among choices that can be made one can find:

1. Nodes of a derivation tree are either single formulae or else sets of formulae,
2. Formulae are accompanied or not by labels conveying semantical informations. Typically, in the case of modal logics, such labels – as in the case for instance of *prefixed tableaux*– allow for an explicit representation of the

semantical transition relation $\mathcal{R}$. When labels are absent, the nature of the transition relation is implicitly encoded into the rule formulation.

Here we have chosen to formulate modal tableaux as trees of finite sets of formulae and to use the implicit representation of transitions.

Moreover, independently from item 2 above, tableaux can be either *signed* or *unsigned*. Here we will shortly present both formats.

No matter which the format is, generally tableau calculi are *refutation systems*: to prove the validity of a formula φ one tries to build a tableau that, intuitively, shows that the hypothesis of falsity of φ leads to contradiction. The tableau itself is seen as systematic search for a possible model of $\neg\varphi$, and when such a search fails $\neg\varphi$ is declared insatisfiable.

If one chooses sets of formulae as the data structures to which the rules of the sequent calculus apply (which is indeed possible, as we observed), and ignores the consequent context dependent formulation of sequent rules for the logical operators, then it becomes apparent that the structure of a tableau is quite similar similar to the structure of a sequent derivation. A tableau for a set of formulae Γ (signed or not, see below) is a finite tree such that: each node is a set of formulae, the root is Γ, and the children of a node Δ are the sets of formulae $\Delta_1, ..., \Delta_n$ such that $\frac{\Delta}{\Delta_1,...,\Delta_n}$, $n \geq 1$ is an instance of one of the rules of the calculus. The rules are usually called *expansion rules* and $\Delta_1, ..., \Delta_n$ are called *expansions* of Δ. In the case of tableaux trees are usually written with the root up and the leaves down. When the leaves are all *closed* (showing a contradiction) we have a refutation of Γ. We postpone the formal definition notion of "closed node", because it slightly differs according to the signed or unsigned format.

Signed Tableaux for S4. The content of this section is a variant of Melvin Fitting's description of signed modal tableaux in his introduction chapter of [DGHP99].

A *signed formula* is a couple $\langle \varphi, pol \rangle$ where φ is a modal formula (according to the grammar of Section 2.1) and $pol \in \{T, F\}$ is a *polarity*. The semantics-oriented intended meaning of the polarities is *true* (T) and *False* (F), hence $\langle \varphi, T \rangle$ can be read as "φ is true" while $\langle \varphi, F \rangle$ can be read as "φ is false".

Signed formulae that are not atomic are classified as α-*formulae* (or conjunctive formulae), β-*formulae* (or disjunctive formulae), ν-formulae and π-formulae, having, respectively, α-*components*, β components, ν_0 *components* and π_0-*components* as indicated in Figures 3 and 4. Observe that, reading the polarity T as "is true" and F as "is false", any α signed formula holds if and only if both the components α_1 et α_2 hold, while any β signed formula holds if and only if at least one of the components β_1 et β_2 hold.

The expansion rules use the analysis of (signed) formulae in the four classes and their respective components. The rule schemata are given in Figure 5. For each rule but (β) the set of formulae below the inference line is called the *expansion*; the rule (β) creates a branching in the tree and has two expansions.

A node is *closed* whenever it contains a pair of signed formulae of the form $\langle \psi, T \rangle$, $\langle \psi, F \rangle$. A *closed tableau* is a finite tableau whose leaves are all closed.

α	α_1	α_2
$\langle \varphi \wedge \psi, T\rangle$	$\langle \varphi, T\rangle$	$\langle \psi, T\rangle$
$\langle \varphi \vee \psi, F\rangle$	$\langle \varphi, F\rangle$	$\langle \psi, F\rangle$
$\langle \neg\varphi, T\rangle$	$\langle \varphi, F\rangle$	$\langle \varphi, F\rangle$
$\langle \neg\varphi, F\rangle$	$\langle \varphi, T\rangle$	$\langle \varphi, T\rangle$

β	β_1	β_2
$\langle \varphi \wedge \psi, F\rangle$	$\langle \varphi, F\rangle$	$\langle \psi, F\rangle$
$\langle \varphi \vee \psi, T\rangle$	$\langle \varphi, T\rangle$	$\langle \psi, T\rangle$

Fig. 3. α and β formulae and their components : signed version

ν	ν_0
$\langle \Box\varphi, T\rangle$	$\langle \varphi, T\rangle$
$\langle \Diamond\varphi, F\rangle$	$\langle \varphi, F\rangle$

π	π_0
$\langle \Diamond\varphi, T\rangle$	$\langle \varphi, T\rangle$
$\langle \Box\varphi, F\rangle$	$\langle \varphi, F\rangle$

Fig. 4. ν and π formulae and their components : signed version

$\dfrac{\Gamma, \alpha}{\Gamma, \alpha_1, \alpha_2}\ (\alpha)$	$\dfrac{\Gamma, \beta}{\Gamma, \beta_1 \quad \Gamma, \beta_2}\ (\beta)$
$\dfrac{\Gamma, \nu}{\Gamma, \nu_0, \nu}\ (\nu)$	$\dfrac{\Gamma, \pi}{\Gamma\#, \pi_0}\ (\pi) \qquad where\ \ \Gamma\# = \{\nu \mid \nu \in \Gamma\}$

Fig. 5. S4 Tableau rules

A closed tableau whose root has the form $\langle \varphi, T\rangle$ is a *refutation* of the unsigned formula φ, thus a proof of $\neg\varphi$'s validity.

Under a semantical reading the intuitive meaning of all rules but the π rule is: "If the premise holds at a world w of an ITS then at least one of the expansions holds at w". The intuitive meaning of the π rule is: "If the premise holds at a world w of an ITS then the expansion holds at some world w' accessible from w. So, we observe that all rules but π are "statical": their role is to analyse what means for a formula φ to be true or false (according to the polarity) at a given world w, while the π rule is "dynamical": it makes a move from the current world to an accessible world. A syntactical counterpart of this feature is that all rules but the π rule are *reversible* in the sense that there is a closed tableau for the premise if and only if there are closed tableaux for the expansions.

Below we prove again by refutation the validity of the formula ϕ=$(\Box p) \rightarrow \Box\Box p$. This time we use the signed tableau rules to build a closed tableau for $\langle (\Box p) \wedge \Diamond\Diamond\neg p, T\rangle$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle(\Box p)\wedge(\Diamond\Diamond\neg p),T\rangle}{\langle\Box p,T\rangle,\langle\Diamond\Diamond\neg p,T\rangle}\,(\alpha)}{\langle\Box p,T\rangle,\langle p,T\rangle,\langle\Diamond\Diamond\neg p,T\rangle}\,(\nu)}{\langle\Box p,T\rangle,\langle\Diamond\neg p,T\rangle}\,(\pi)}{\langle\Box p,T\rangle,\langle p,T\rangle,\langle\Diamond\neg p,T\rangle}\,(\nu)}{\langle\Box p,T\rangle,\langle\neg p,T\rangle}\,(\pi)}{\langle\Box p,T\rangle,\langle p,T\rangle\langle\neg p,T\rangle}\,(\nu)}{\langle\Box p,T\rangle,\langle p,T\rangle\langle p,F\rangle}\,(\alpha)$$

The example shows very clearly a general phenomenon that could have been already remarked by inspecting rules: if one reads the polarities not semantically but as "left" (T) and "right" (F) – the two sides of a sequent –, then there is a close correspondence between the rules of the signed tableaux for $\mathsf{S4}$ as given in Figure 5 and those of the two sided sequent calculus for $\mathsf{S4}$, as given in the Figures 1 and 2. The rules (α) and (β) are a syntactic variant of the sequent rules for the boolean operators, while the (ν) rule corresponds to the sequent rules $\Box L$ and $\Box R$ and the (π) rule to the sequent rules $\Diamond L$ and $\Diamond R$; traditionally tableau rules (and inferences) are written upside down with respect to sequent rules.

3.3 Left-handed Sequent Calculi and Unsigned Tableaux for S4

We have observed that formulae can be always rewritten in fnn. This allows for a one-sided formulation of the sequent rules and for an unsigned version of the tableau rules that lead literally to the same calculus.

For the sequent calculus, it suffices to:

- Consider only sequents having an empty succedent: $\Gamma \Rightarrow$, where elements of Γ are in fnn,
- Formulate axiom schemas as: $\overline{\Gamma, p, \neg p \Rightarrow}$ where p is atomic,
- Consider only left rules among the rules in Figures 1 and 2,
- Suppress the rules for negation.

It is worthwhile noticing that in the obtained left-handed calculus, that we note here $\mathsf{S4}_{LSeq}$, only sequents having an associated formula $\Gamma_\varphi \rightarrow \bot$ can be proved, thereby refuting Γ_φ and showing only indirectly that $\neg\Gamma_\varphi$ is valid.

For the tableau calculus, a corresponding unsigned version can be obtained by considering only sets of formulae in fnn whose polarity is T, that can therefore be omitted because useless. Decomposition of unsigned formulae can be defined as in Figure 6; observe that negated formulae do not appear there. Modulo such a reading of decompositions, the formulation of the tableau rules remain the same as in the signed case. For the obtained tableau calculus, that we call here $\mathsf{S4}_{LTab}$, a node is declared close whenever it contains both $\langle p\rangle$ and $\langle\neg p\rangle$ for some propositional letter p.

Not only it is true that, for any formula φ, there is a proof of the sequent $\varphi \Rightarrow$ if and only if there is a closed tableau for φ, but sequent proofs and tableau refutations are literally the same syntactical object, but for the following trivial features:

- The nodes of a $\mathsf{S4}_{LSeq}$ proof contain the meta-logical symbol $\Rightarrow$ while those of a $\mathsf{S4}_{LTab}$ proof do not; however such a symbol is useless in a one-sided context, and it can be removed from sequents,
- For just historical conventional reasons, the root of a $\mathsf{S4}_{LSeq}$ proof is at the bottom, while in a tableau it is at the top.

Therefore tableaux for $\mathsf{S4}$ can be seen as derivations in a left handed sequent calculus whose rules are formulate so as to minimize non determinism in a proof search that starts from the root. This holds also for K, T, $\mathsf{K4}$ and some other modal logics.

α	α_1	α_2
$\varphi \wedge \psi$	φ	ψ

β	β_1	β_2
$\varphi \vee \psi$	φ	ψ

ν	ν_0
$\Box\varphi$	φ

π	π_0
$\Diamond\varphi$	φ

Fig. 6. Formulae and their components: unsigned version

4 Proofs for LTL

The logic $\mathsf{S4}$ allows for an unsatisfactory representation of time, since a $\mathsf{S4}$ frame may contain cycles, while LTL provides a more faithful representation. However the straightforward link existing between sequent and tableau derivations in the case of $\mathsf{S4}$ is not any longer available in the case of LTL (and other temporal logics like CTL or CTL^*). This can be explained with the fact that for LTL the fixed point equivalences seen in Section 2.2 hold, and all the istances of the induction schema are valid. As a consequence, the so called *eventualities*, namely formulae of the form $\varphi \mathsf{U} \psi$ or $\Diamond\psi$, that "promise" to eventually make ψ true, are source of specific difficulties. In fact if, for instance, a sequent or tableau rule for $\Diamond$ is designed so to mirror the equivalence $\Diamond\psi \equiv \psi \vee \bigcirc\Diamond\psi$, then one needs to check that the realization of ψ is not delayed *ad libitum* while trying to build a model through tableau construction to conclude that the root is satisfiable. This can happen when only the right disjunct in of $\psi \vee \bigcirc\Diamond\psi$ is always available, other paths containing only inconsistent vertices having been suppressed (see later on). Therefore the need arises to check a global condition on the derivation in order to conclude about the satisfiability of the root.

4.1 Two-pass tableaux for LTL: Wolper's tableaux

A pioneer calculus to reason on the satisfiability/validity of LTL formulae has been proposed by P. Wolper in [Wol85]. In order to check the satisfiability of a set of formulae Γ one proceeds in two phases:

1. Construction phase: a directed graph (not a tree!) of sets of LTL formulae rooted at Γ is built, using appropriate expansion rules. When a consistent vertex v is expanded, and the application of an expansion rule would create as v's successor a second copy of a vertex v' already existing, then v' is not duplicated but rather an edge from v to v' is added. The construction phase always terminates, generating a finite graph, say G.
2. Elimination Phase: "bad vertices" of G, that cannot contribute to the construction of a model of Γ are recursively suppressed. If the root Γ survives then it is declared satisfiable, otherwise it is declared unsatisfiable.

We give here a presentation that is a slight variation of the material in [Wol85]. We have chosen here to work with formulae which are in fnn. A formula which is either a literal or has the form $\bigcirc\varphi$ is called *primitive* (or *elementary*). Non-primive formulae can be classified as α or β formulae as indicated in Figure 7; as usual α formulae have a conjunctive nature while β formulae are disjunctive. Let us observe that here we have included $\Box\varphi$ in the class of α-formulae, and $\varphi \mathsf{U} \psi$ and $\Diamond\varphi$ in the class of β-formulae, as it is traditionally done.

α	α_1	α_2
$\varphi \wedge \psi$	φ	ψ
$\Box\varphi$	φ	$\bigcirc \Box\varphi$

β	β_1	β_2
$\varphi \vee \psi$	φ	ψ
$\varphi \mathsf{U} \psi$	ψ	$\varphi \wedge \bigcirc \varphi \mathsf{U} \psi$
$\Diamond\varphi$	φ	$\bigcirc \Diamond\varphi$

Fig. 7. α and β formulae for LTL

On can observe that the decompositions of $\Box\varphi$, $\Diamond\varphi$ and $\varphi \mathsf{U} \psi$ are founded on the fixpont equivalences for LTL seen in Section 2.2.

Say that a set of LTL formulae is *patently inconsistent* if it contains both p and $\neg p$ for some $p \in P$.

There are two expansion rules: a static one, that we call here SR, and a dynamic one, $Next$.

***SR* Rule.** Let Γ be a set of formulae that is not patently inconsistent – let us say that it is "consistent" for short. A set of formulae Δ is *full expansion* of Γ if it is the result of the following iterative procedure (that obviously halts):

- Initialisation: Set $\Delta := \Gamma$
- Saturation:

- (α-Saturation). If $\varphi \in \Delta$ is an α-formula and it is not marked as "used", then add to Δ both its components and mark φ as used;
- (β-Saturation). If $\varphi \in \Delta$ is a β-formula and it is not marked as "used", then add to Δ one of its components and mark φ as used.

It is worthwhile noticing that the above procedure is non-deterministic, thus several full expansions of Γ can be produced. Let us note $FS(\Gamma)$ the family of all the full saturations of Γ. The rule SR expands a vertex Γ that is consistent and contains at least a non-primitive formula by creating all the elements of $FS(\Gamma)$ as Γ's successors (but avoiding duplications, as said above). A vertex created as the result of the application of the SR-rule is said to be a *tableau state*.[5]

For instance, if the current node Γ is $\Box\Diamond p, (q \wedge p)\mathsf{U}r$, four successors of Γ are created:
$\Delta_1 = \Box\Diamond p, (q \wedge p)\mathsf{U}r, \Diamond p, \bigcirc\Box\Diamond p, p, r$
$\Delta_2 = \Box\Diamond p, (q \wedge p)\mathsf{U}r, \Diamond p, \bigcirc\Box\Diamond p, \bigcirc\Diamond p, r$
$\Delta_3 = \Box\Diamond p, (q \wedge p)\mathsf{U}r, \Diamond p, \bigcirc\Box\Diamond p, p, (q \wedge p) \wedge \bigcirc((q \wedge p)\mathsf{U}r), (q \wedge p), q, p, \bigcirc((q \wedge p)\mathsf{U}r)$
$\Delta_4 = \Box\Diamond p, (q \wedge p)\mathsf{U}r, \Diamond p, \bigcirc\Box\Diamond p, \bigcirc\Diamond p, (q \wedge p) \wedge \bigcirc((q \wedge p)\mathsf{U}r), (q \wedge p), q, p, \bigcirc((q \wedge p)\mathsf{U}r)$

It is worthwhile observing that the SR rule is conservative, in the sense that the decomposed formula is kept in the result of the decomposition, taking care –however– of not analyzing it twice, by means of a suitable marking device. This is different from what happens in the tableaux for $\mathsf{S4}$ that we saw, and it is necessary because of the elimination phase, that needs to memorize any possible eventuality in order to assure that its fulfillment is not procrastinated *ad libitum* (see below). One may also remark that this formulation encompasses a sequence of applications of the (α) and (β) of $S4_{LTab}$ into one single step.

***Next* Rule.** The $Next$ rule applies only to tableau states, namely sets of formulae that are fully expanded. Let $L_1, ..., L_n, \bigcirc\varphi_1, ..., \bigcirc\varphi_m$, where $n + m \geq 1$ and each L_i is a literal be all the primitive formulae in a state Δ. The $Next$ rule creates a successor node $\Gamma' = \varphi_1, ..., \varphi_m$. In the particular case where $m = 0$, $\Gamma' = \top$. A node Γ' containing just $\top$ is also considered a tableau state (and a new application of $Next$ to Γ' creates a loop).

Once a tableau G is built, vertices are recursively suppressed in the elimination phase by iterating the following operations as much as it is possible:

1. Remove any vertex who has no successors (thus, in particular, any patently inconsistent vertex),
2. Remove any vertex that contains as an element an eventuality $\Diamond\psi$ or $\varphi\mathsf{U}\psi$ and that is not the origin of any path in the current graph leading to a node containing ψ. The role of this operation is, intuitively, to eliminate as possible models those interpretations where an eventuality should hold at a

[5] Let us observe that the word 'state" is somewhat semantically overloaded, since it has a meaning in the context of the semantics of LTL and a meaning here, as a kind of vertex in a tableau. However the two notions have an intuitive relation, since a tableau state actually is a representation of a state according to the first sense.

state but the promised formula is never obtained. It is important to observe that checking that an eventuality occurring in some state is fulfilled via at least a path is a *global condition* on the current tableau.

In this calculus a tableau is closed whenever at the end of the elimination phase its root Γ is suppressed; in that case Γ is refuted, and declared unsatisfiable.

When a tableau is complete, that is all vertices have been expanded, but it is *open* (*i.e.* not closed), then it is always possible to build out of the tableau an interpretation satifying its root. Thus the calculus is sound and a complete with respect to the unsatisfiability: Γ is unsatisfiable if and only if there is a closed tableau rooted at Γ. Tableaux provide a constructive decision method for the satifiability problem, as well as for the validity problem. Both the satisfiability and the validity problems for LTL are PSPACE complete [DGL16], thus the complexity of such a decision procedure is sub-optimal, because it is exponential in the size of the argument formula. This is due to the fact that the number of vertices of a tableau rooted at φ can be 2^m, where m is the size of φ.

Clearly the structure of this calculus is far from standard sequent calculi: it makes an essential use of loop-check in the very definition of what a tableau is, it works in two phases and it needs a global condition to determine when a tableau is indeed a (refutational) proof. Tableaux for LTL rather recall the construction of Büchi automata to decide LTL satisfiability (see for instance [Pel01]).

4.2 Sequent calculi and one-pass tableaux and for LTL

Some early works on sequent calculi for temporal logics present two sorts of problems that make them inappropriate to do proof search. Some of them, for instance [Kaw87], contain an infinitary rule:

$$\frac{\Rightarrow \Gamma, A \qquad \Rightarrow \Gamma, \bigcirc A \qquad \Rightarrow \Gamma, \bigcirc\bigcirc A \quad ...}{\Rightarrow \Gamma, \Box A}$$

Others, as the one in [Pae88] for instance, are not cut-free since they contain this kind of rule:

$$\frac{\Rightarrow \Gamma, B \qquad \Rightarrow \neg B, \bigcirc B \qquad \Rightarrow \neg B, A}{\Rightarrow \Gamma, \Box A}$$

These difficulties are clearly related to the specific feature of temporal linear logics w.r.t. other modal logics that we already mentioned: the capability of validating induction, so to encode $\mathbb{N}$.

However, some one-pass tableau calculi for LTL where there are not two distinct phases have been proposed, after Wolper's seminal work, and they sometimes correspond to sequent systems that are cut-free and finitary. We give here a short survey of these works, treating at the same time one-pass tableaux and sequent-calculi (since it is sometimes difficult to separate the two approaches in

these works), without any claim of exhaustiveness (in particular with respect to works on μ-calculus).

In [Sch98] tableaux for LTL are tree-shaped, however there are some communications between branches. There are also some annotations of nodes with depth measures. The tableau calculus there defined is one-pass and checks on the fly, branch by branch, the fulfillment of eventualities.

Cut free sequent systems for LTL and CTL are proposed in [BL08]. Also here a proof is a finite tree, that uses annotated formulae (not belonging to the object logical language). It can be observed that a loop rule is present which hides a non-local closing condition, because it is necessary to inspect previous nodes in order to verify if there is a loop.

The authors of [GHL$^+$09] provide a one-phase tableau method for LTL (called PTL there), named TTM, that is associated to a left-handed cut-free sequent calculus TTC. This, in its turn, is transformed in a two-sided sequent calculus GTC that is finitary, sound and complete for LTL. Therefore, the authors claim to have shown that 'the classical correspondence between tableaux and sequent calculi can be extended to TL" (Temporal Logics). However they acknowledge that their approach has a poor performance as a decision method for the LTL satisfiability problem, because the structural complexity of the problem is PSPACE, traditional two phases tableaux require exponential time and their own approach requires even double-exponential time (in the worst case).

In [Rey16] a tableau calculus for LTL is defined where a tableau is, again, a tree of sets of formulae, not a graph. The main novelty is the definition of a new tableau rule, called PRUNE, allowing for the finite detection of any branch that, even if continued *ad libitum*, would not produce any new fulfillment of an eventuality. However the same remark done for [BL08] holds for this work: one needs to explore some ancestors of the current node in order to apply PRUNE. This approach has been extended to an extension of LTL to past time in [GMR17].

Although the work [BN09] is older than [Rey16], we shortly discuss it after the above articles because it uses a quite specific approach to the definition of sequent calculi, namely a *labelled sequents* approach, as previously introduced by the second author in [Neg05]. The use of labels, in general, allows for embedding model-theoretic arguments in the syntax of the inference rules. The studied logic in [BN09] is Prior's Linear Time Logic, a modal temporal logic logic introduced by Arthur Prior in the late 1950s and containing also past operators. However, LTL is nothing but the future-oriented and *reflexive* version of Prior's Linear Time Logic; for instance, in Prior's logic $\Box\varphi$ means that in the strict future - not necessarily including the present - φ will always hold. In [BN09] eventualities have only the form $\Diamond\phi$, because U (and its past corresponding operator, *since*) is not taken into account, however this is done in the first author's Ph.D. thesis [Bor08]. Candidate proofs in the calculus – named G3LT$_{cl}$ – are possibly infinite trees of labelled sequents. The calculus G3LT$_{cl}$ is sound and complete with respect to validity, and an appropriate proof search procedure always terminates, notwithstanding the fact that a tree having a not valid root might be infinite, in principle. Hence, G3LT$_{cl}$ provides a decision algorithm for validity.

It is worthwhile observing that the proof search method makes no loop-check and no inspection of previous parts of the tree: at any step of the construction of a candidate proof only the current sequent in inspected, possibly allowing for halting the search. Thus the halting conditions are local. In particular, one can stop the construction of the current branch because a sequent $\mathcal{S}$ that is an axiom is found – or else, because a so called *fulfilling sequent* $\mathcal{S}$ is found, providing a counter-model of the root sequent, and in this second case the root sequent is declared invalid. A practical drawback of this calculus, however, is the fact that, as [GHL+09], it provides a decision algorithm for validity that has a suboptimal complexity.

Finally, LTL validities can be proved by using the more expressive linear time μ-calculus, noted L_μ^{Lin}, that extends LTL with greatest and smallest fixed point operators. This gives it the power to express all Ω-regular languages, i.e. strictly more than LTL. The linear time μ-calculus can also be seen as the modal μ-calculus interpreted only over infinite linear time structures, i.e. Kripke structures in which every state has exactly one successor. The structural complexity of the validity problem for L_μ^{Lin} formulae is PSPACE, as for LTL. However, the presence of nested and possibly alternating fixed point constructs makes its validity decision problem much harder than for LTL, in practice.

In the work [DHL06] the authors in propose a sequent calculus for L_μ^{Lin} where a proof is an infinite tree in which each branch satisfies an additional global condition concerning the existence of so-called *threads*. Although proofs are infinite objects, the combination of the use of the calculus with, respectively, automata tools and category-theoretic methods allows for the definition of two algorithms deciding validity.

The logic L_μ^{Lin} is studied also in [DBHS16]. More precisely, three calculi are considered: the infinitary system μLK^∞, its regular fragment μLK^Ω and the finitary sequent calculus μLK. Constructive completeness results are provided for class of formulae corresponding to interesting fragments of L_μ^{Lin}. As the authors argue, a constructive proof of completeness (for instance, specifying a proof search method) readily provides a realistic decision algorithm.

In [JKS08] two infinitary cut frees sequent calculi are defined for the modal μ-calculus (thus not for the specific L_μ^{Lin}). However, such calculi are then transformed into a finitary one by exploiting the small model property of the propositional modal μ-calculus. Hence, the finite model property is presupposed by the definition itself of the calculus rather then being a consequence of soundness, validity and termination, as it is often case for sequent or tableau based formalizations of logics.

The work [AL17] present two sound and complete finitary cut-free sequent calculi for the modal μ-logic.

One of the authors of [JKS08] contributes also to the more recent work [AJL19] that extends the study to the modal μ-calculus with converse modalities; this extended logic is known as *two-way μ-calculus* or *full μ-calculus*.

5 Alternating Time Temporal Logic

Temporal logics of the ATL family, first appearing in [AHK97,AHK02], consider several agents who act synchronously. They have been proposed to model the behavior of so called "open" computer systems – where the environment is unknown or only partially known – as a game between the system components and an "enemy", the environment. The semantical scenarios of ATL logics are in fact games where agents can form coalitions in order to achieve a given goal. One can express proprieties as "Coalition C has a strategy to ensure achievement of the goal g, no matter how the other agents play". Intuitively, these logics are extensions of computer tree logics to the multi-agent case.

We recall here some standard definitions about ATL*.

Definition 4 (Concurrent Game Model). *Given a set of* atomic propositions *P, a CGM (Concurrent Game Model) is a 5-tuple*

$$\mathcal{M} = \langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_a\}_{a\in\mathbb{A}}, \{\mathsf{act}_a\}_{a\in\mathbb{A}}, \mathsf{out}, \mathsf{L}\rangle$$

such that:

- $\mathbb{A} = \{1, ..., k\}$ *is a finite non-empty set of agents,*
- $\mathbb{S}$ *is a non-empty set of states,*
- *For each $a \in \mathbb{A}$, Act_a is a non-empty set of actions. If $A \subseteq \mathbb{A}$, then A is a coalition of agents. Given a coalition A, an A-move is a k-ple $\langle \alpha_1, ..., \alpha_k\rangle$ where, for any $i, 1 \leq i \leq k$, if $i \in A$ then $\alpha_i \in \mathsf{Act}_a$, else $\alpha_i = *$ ($*$ being a place-holder symbol distinct from each action). A move of the coalition $\mathbb{A}$ will also be called global move. The set of all the A-moves is denoted by Act_A. The notation σ_A denotes an element of Act_A, and if $a \in A$, $\sigma_A(a)$ means the action of the agent a in the A-move σ_A,*
- act_a *is a function mapping a state s to a non-empty subset of Act_a; $\mathsf{act}_a(s)$ denotes the set of actions of the agent a that are available at state s. Given a coalition A, a mapping act_A associating to a state a set of A-moves is naturally induced by the function act_a; $\mathsf{act}_A(s)$ is the set of all the A-moves available to coalition A at state s,*
- out *is a transition function, associating to each $s \in \mathbb{S}$ and each $\sigma_{\mathbb{A}} \in \mathsf{act}_{\mathbb{A}}(s)$ a state $\mathsf{out}(s, \sigma_{\mathbb{A}}) \in \mathbb{S}$: the state reached when each $a \in \mathbb{A}$ does the action $\sigma_{\mathbb{A}}(a)$ at s,*
- L *is a labelling function $\mathsf{L} : \mathbb{S} \to \mathcal{P}(P)$, associating to each state s the set of propositions holding at s.*

5.1 ATL* Syntax and Semantics

Let A be a *coalition* of agents. The following grammar defines formulae of ATL*, the most expressive logic of the ATL family, by mutual recursion.

Definition 5 (ATL* syntax).

State formulae :
$\psi := a \mid \neg(\psi) \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid (\langle\langle A\rangle\rangle \Phi) \mid ([[A]]\Phi)$
where a is an atomic formula.

Path formulae :
$\Phi := \psi \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\bigcirc\Phi) \mid (\Box\Phi) \mid (\Phi \mathsf{U} \Phi)$

The expressions $\langle\langle A\rangle\rangle$ and $[[A]]$ are called *strategic quantifiers*. It is worthwhile observing that any ATL^* state formula is also an ATL^* path formula, while the converse is false. State formulae will always be noted by lower case Greek letters, and path formulae by upper case Greek letters[6]. Unless explicitly stated otherwise, in the sequel by ATL^* formula we mean an ATL^* state formula.

The logic called simply ATL, or *Vanilla* ATL to avoid confusions with the family, is the syntactical fragment of ATL^* obeying to the constraint that any temporal operator in a formula is prefixed by a quantifier and that no quantifier can have a boolean or temporal operator in its immediate scope, analogously to CTL w.r.t. CTL^*. Hence any ATL formula is a state formula. The intermediate logic ATL^+ does not allow for nesting temporal operators in path formulae but allows for boolean combinations of path formulae in the immediate scope of a quantifier.

The semantics for ATL^* is based on the notions of concurrent game model, *play* and *strategy*.

A play λ in a CGM $\mathcal{M}$ is an infinite sequence of elements of $\mathbb{S}$: $s_0, s_1, s_2, ...$ such that for every $i \geq 0$, there is a global move $\sigma_{\mathbb{A}} \in \mathsf{act}_{\mathbb{A}}(s_i)$ such that $\mathsf{out}(s_i, \sigma_{\mathbb{A}}) = s_{i+1}$. Given a play λ, we denote by λ_0 its initial state, by λ_i its $(i+1)$th state, by $\lambda_{\leq i}$ the prefix $\lambda_0...\lambda_i$ of λ and by $\lambda_{\geq i}$ the suffix $\lambda_i\lambda_{i+1}...$ of λ. Given a prefix $\lambda_{\leq i} : \lambda_0...\lambda_i$, we say that it has length $i+1$ and write $|\lambda_{\leq i}| = i+1$. An empty prefix has length 0. A (non-empty) *history* at state s is a finite prefix of a play ending with s. We denote by $\mathsf{Plays}_{\mathcal{M}}$ and $\mathsf{Hist}_{\mathcal{M}}$ respectively the set of plays and set of histories in a CGM $\mathcal{M}$.

Given a coalition $A \subseteq \mathbb{A}$ of agents, a *perfect recall A-strategy* F_A is a function which maps each element $\lambda = \lambda_0...\lambda_\ell$ of $\mathsf{Hist}_{\mathcal{M}}$ to an A-move σ_A belonging to $\mathsf{act}_A(\lambda_\ell)$ (the set of actions available to A at state λ_ℓ).

Given a coalition $A \subseteq \mathbb{A}$ of agents, a *perfect recall A-co-strategy* F^c_A is a function which assigns to every A-strategy and each element $\lambda = \lambda_0...\lambda_\ell$ of $\mathsf{Hist}_{\mathcal{M}}$ a collective move of $A^c = \mathbb{A} \setminus A$ (the coalition of A's opponents) belonging to $\mathsf{act}_{A^c}(\lambda_\ell)$ (the set of actions available to A^c at state λ_ℓ). Intuitively, we can see A^c as a way of answering to A-actions by the opponents of A.

Whenever a strategy (or a co-strategy) depends only on the last state λ_ℓ of the history $\lambda = \lambda_0...\lambda_\ell$ it is said to be *positional* (or *memoryless*).

For any coalition A, a global move $\sigma_{\mathbb{A}}$ *extends* an A-move σ_A whenever for each agent $a \in A$, $\sigma_A(a) = \sigma_{\mathbb{A}}(a)$. Let σ_A be an A-move; the notation $\mathsf{Out}(s, \sigma_A)$ denotes the set of states $\mathsf{out}(s, \sigma_{\mathbb{A}})$ where $\sigma_{\mathbb{A}}$ is any global move extending σ_A. Intuitively, $\mathsf{Out}(s, \sigma_A)$ denotes the set of the states that are successors of s when

[6] Again, the meaning of any given occurrence of a Greek capital letter – a path formula or a set of formulae – will be clear from the context.

the coalitions A plays at s the A-move σ_A and the other agents play no matter which move.

A play $\lambda = \lambda_0, \lambda_1, ...$ is said to be *compliant with a strategy (respectively: a co-strategy)* F_A if and only if for each $j \geq 0$, $\lambda_{j+1} \in \mathsf{Out}(\lambda_j, \sigma_A)$, where σ_A is the A-move chosen by F_A at state λ_i (given also the current history for the perfect recall case).

The notions $\mathcal{M}$ *satisfies the formula* Φ *at state* s, noted $\mathcal{M}, s \models \Phi$, and $\mathcal{M}$ *satisfies the formula* Φ *at path* λ, noted $\mathcal{M}, \lambda \models \Phi$, are recursively defined as follows (omitting the obvious boolean cases):

- $\mathcal{M}, s \models \langle\langle A \rangle\rangle \Phi$ iff there exists an A-strategy F_A such that, for all plays λ starting at s and compliant with the strategy F_A, $\mathcal{M}, \lambda \models \Phi$,
- $\mathcal{M}, s \models [[A]]\Phi$ iff there exists an A-co-strategy F_A^c such that, for all plays λ starting at s and compliant with F_A^c, $\mathcal{M}, \lambda \models \Phi$,
- $\mathcal{M}, \lambda \models \varphi$ iff $\mathcal{M}, \lambda_0 \models \varphi$,
- $\mathcal{M}, \lambda \models \bigcirc \Phi$ iff $\mathcal{M}, \lambda_{\geq 1} \models \Phi$,
- $\mathcal{M}, \lambda \models \Box \Phi$ iff $\mathcal{M}, \lambda_{\geq i} \models \Phi$ for all $i \geq 0$;
- $\mathcal{M}, \lambda \models \Phi \mathsf{U} \Psi$ iff there exists an $i \geq 0$ where $\mathcal{M}, \lambda_{\geq i} \models \Psi$ and for all $0 \leq j < i$, $\mathcal{M}, \lambda_{\geq j} \models \Phi$.

Given a CGM $\mathcal{M}$, a state s of $\mathcal{M}$ and a formula ϕ, we say that that ϕ *is true at* s in $\mathcal{M}$ whenever $\mathcal{M}, s \models \phi$. We say that a CGM $\mathcal{M}$ is a *model* of ϕ whenever there is a state s such that ϕ is true at s in $\mathcal{M}$, and is a model of a finite set of state formulae Γ whenever it is a model of the conjunction of the elements of Γ.

The two kinds of strategies, positional and perfect recall, produce incomparable sets of satisfiable/valid formulae [JB11]. In the rest of the paper we always consider perfect recall strategies. Also, one can consider two notions of satisfiability of φ: one can set that the relevant agents to be considered are only those explicitly occurring in φ – then one speaks of "tight satisfiability" - or else one can consider a larger set of agents. Here we consider tight satisfiability.

A path formula Φ is a *logical consequence* of a path formula Ψ, noted $\Psi \models \Phi$, when for any CGM $\mathcal{M}$ and any path λ, $\mathcal{M}, \lambda \models \Psi$ implies $\mathcal{M}, \lambda \models \Phi$. The formulae Ψ and Φ are said to be *equivalent*, noted $\Psi \equiv \Phi$ when $\Psi \models \Phi$ and $\Phi \models \Psi$.

The following fixed-point equivalences hold for ATL^* with perfect recall strategies:

- $\langle\langle A \rangle\rangle \Box \Phi \equiv \Phi \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box \Phi$.
- $\langle\langle A \rangle\rangle \Phi \mathsf{U} \Psi \equiv \Psi \vee (\Phi \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Phi \mathsf{U} \psi)$.
- The analogous of the equivalences above with $[[A]]$ replacing $\langle\langle A \rangle\rangle$.

The path formula $\Diamond \Phi$ can be defined by $T \mathsf{U} \Phi$, and $\Phi \mathsf{R} \Psi$ (Φ releases Ψ) can be defined by $(\Box \Psi) \vee (\Phi \mathsf{U} (\Phi \wedge \Psi))$. Since $\neg \langle\langle A \rangle\rangle \Phi \equiv [[A]] \neg \Phi$ and $\neg [[A]] \Phi \equiv \langle\langle A \rangle\rangle \neg \Phi$ it is easy to see that ATL^* formulae can be rewritten in fnn.

6 Tableaux for deciding ATL* satisfiability

In [Dav15a] a tableau system is defined that decides the problem of ATL* state formulae (tight, perfect-recall) satisfiability, answering to the question: "Given any formula ϕ are there a CGM $\mathcal{M}$ and a state s such that $\mathcal{M}, s \models \Phi$?" Such a work extends to ATL* the tableau calculus proposed in [CDG14,CDG15] for ATL$^+$. The calculus is a two-pass tableau system; up to our knowledge, it is the only calculus deciding ATL* satisfiability problem, that can also be solved, however, by automata tools [Sch08].

It is worthwhile emphasizing that tableau construction, here, should be understood as a systematic search of CGMs that are models of the root, as for LTL. In the case of failure to find such a model the tableau is declared closed and its root unsatisfiable.

Again as in the case of LTL, a tableau is a finite graph of formulae, more precisely of state formulae, built – avoiding duplications of vertices – by using a static rule and a dynamic rule, and then modified by eliminating undesirable vertices. Also here the static rule uses formulae decomposition, and such a decomposition is founded on fixed point equivalences. Such a decomposition, however, and the corresponding classification of formulae is quite specific to ATL*, and to the discussed calculus.

Without loss of generality we consider only formulae in fnn.

6.1 Construction Phase

Classification and decomposition of ATL* state (fnn) formulae. Let A be a coalition of agents and let us note Q a strategy quantifier, namely $\langle\langle A\rangle\rangle$ or $[[A]]$.

A *successor formula* is a formula having either the form $\langle\langle A\rangle\rangle \bigcirc \Phi$ or $[[A]] \bigcirc \Phi$. The class of ATL* state formulae can be partitioned in:

- *Primitive Formulae.* A primitive formula is either a literal or a successor formula $Q \bigcirc \phi$ where ϕ is a state formula, called the successor component of $Q \bigcirc \phi$;
- *Non-primitive formulae* that can be classified as:
  - α*-formulae*, having the form $\phi \wedge \psi$; as before, ϕ and ψ are said to be its α-components;
  - β*-formulae*, having the form $\phi \vee \psi$; again, ϕ and ψ are said to be its β-components;
  - γ*-formulae*, that are non-primitive formulae of the form $Q\Phi$.

The notion of γ-formula has been introduced in [CDG14,CDG15]. The analysis of such formulae in components that are *state formulae*, to be used in the static tableau rule, is delicate, because two difficulties have to be faced:

- A strategic quantifier actually combines first-order existential and universal quantification. For instance, $\langle\langle A\rangle\rangle\Phi$ means that there is a strategy for coalition A such that, no matter how the other agents play, Φ is assured. As a

consequence, any strategic quantifier does not distribute over conjunction and does not distribute over disjunction either.[7]
- When Φ and Ψ are not state formulae, fixed point equivalences cannot be immediately exploited to analyse formulae $Q\Box\Phi$ and $Q\Phi\mathsf{U}\Psi$ into components that are again state formulae. Take, for instance, the γ-formula $\langle\langle A\rangle\rangle\Box\Diamond p$. Its truth at a state s cannot be analyzed as the truth at s of both $\Diamond p$ and $\langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\Diamond p$, because $\Diamond p$ is not a state formula.

However, by iterating the use of ATL^* equivalences, any γ-formula can be indeed analyzed as a disjunction of conjunctions of state formulae. For instance, the truth of γ-formula $\langle\langle A\rangle\rangle((\Box p) \vee (\Box q))$ at a current state s can be analyzed in three cases:

- At present state s, p holds (*present* state formula), and the state formula $\langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle(\Box p)$ holds (*future* state formula); this last express a commitment for the future : for some A-move the paths issued from s's successors and compliant with some given strategy will make the path formula $\Box p$ true;
- At present state s, q holds (*present* state formula), and the state formula $\langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box q$ holds (*future* state formula), expressing the commitment that for some A-move the paths issued from s's successors and compliant with some given strategy will make the path formula $\Box p$ true;
- At present state s, both p and q hold (*present* state formulae), and the state formula $\langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle((\Box p)\vee(\Box q))$ holds (*future* state formula), expressing the commitment that for some A-move the paths issued from s's successors and compliant with some given strategy will make the path formula $(\Box p) \vee (\Box q)$ true. In this last case at the present state no choice has been made yet about which one, among $\Box p$ and $\Box q$, to enforce.

Hence, expansions of a vertex containing the formula $\langle\langle A\rangle\rangle(\Box p) \vee (\Box q))$ are built according to the equivalence:
$\langle\langle A\rangle\rangle(\Box p) \vee (\Box q)) \equiv$
$(p \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box p) \vee (q \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box q) \vee (p \wedge q \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle(\Box(p) \vee \Box(q))$

In the general case, the approach firstly introduced in [CDG14,CDG15], then extended in [Dav15a], analyses γ-formulae by using a function dec which takes as input a path formula and returns a set of triples, whose first element is a state formula, the second is a path formula and the third is a set of path formulae. The definition of dec uses two auxiliary binary functions $\otimes$ and $\oplus$ taking as arguments two sets of triples of this sort and producing a new set of triples. The first element of each triple corresponds to the *present* formula, the second to the commitment for the *future*. The third component does not participate to the decomposition of γ-formulae used in the static expansion rule, but plays a technical role in the elimination phase of the tableau calculus, to check eventuality fulfillment.

The analysis of γ-formulae uses the following definitions. The notations $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to two sets of triples, whose first element is a state formula, the second is a path formula and the third is a set of path formulae, while the Γ_i and

[7] This phenomenon occurs already for the logic ATL^+.

Γ_j are sets of formulae. The operators $\widetilde{\wedge}$ and $\widetilde{\vee}$ correspond, respectively, to the boolean function $\wedge$ and $\vee$ where the associativity, commutativity, idempotence and identity element properties are embedded in the syntax. The aim of both the operators is to automatically generate formulae in conjunctive normal form without redundancy, and ensure the termination of the tableau construction procedure. For instance, $p\widetilde{\wedge}q\widetilde{\wedge}p\widetilde{\wedge}T = p \wedge q = q \wedge p$.

Definition 6. .

- $\mathcal{S}_1 \otimes \mathcal{S}_2 := \{\langle \psi_i \widetilde{\wedge} \psi_j, \Psi_i \widetilde{\wedge} \Psi_j, \Gamma_1 \cup \Gamma_2\rangle \mid \langle \psi_i, \Psi_i, \Gamma_i\rangle \in \mathcal{S}_1, \langle \psi_j, \Psi_j, \Gamma_j\rangle \in \mathcal{S}_2\}$.
- $\mathcal{S}_1 \oplus \mathcal{S}_2 :=$
  $\{\langle \psi_i \widetilde{\wedge} \psi_j, \Psi_i \widetilde{\vee} \Psi_j, \Gamma_1 \cup \Gamma_2\rangle \mid \langle \psi_i, \Psi_i, \Gamma_i\rangle \in \mathcal{S}_1, \langle \psi_j, \Psi_j, \Gamma_j\rangle \in \mathcal{S}_2, \Psi_i \neq \top, \Psi_j \neq \top\}$.

The function dec is defined by recursion on the input path formula Φ.

Definition 7.

- $\mathsf{dec}(\varphi) = \{\langle \varphi, \top, \emptyset\rangle\}$ *for any* ATL^* *state formula* φ.
- $\mathsf{dec}(\bigcirc \Phi_1) = \{\langle \top, \Phi_1, \emptyset\rangle\}$ *for any path formula* Φ_1.
- $\mathsf{dec}(\Box \Phi_1) = \{\langle T, \Box\Phi_1, \{\Phi_1\}\rangle\} \otimes \mathsf{dec}(\Phi_1)$
- $\mathsf{dec}(\Phi_1 \mathsf{U} \Phi_2) = (\{\langle \top, \Phi_1 \mathsf{U} \Phi_2, \{\Phi_1\}\rangle\} \otimes \mathsf{dec}(\Phi_1)) \cup (\{\langle \top, \top, \{\Phi_2\}\rangle\} \otimes \mathsf{dec}(\Phi_2))$.
- $\mathsf{dec}(\Phi_1 \wedge \Phi_2) = \mathsf{dec}(\Phi_1) \otimes \mathsf{dec}(\Phi_2)$
- $\mathsf{dec}(\Phi_1 \vee \Phi_2) = \mathsf{dec}(\Phi_1) \cup \mathsf{dec}(\Phi_2) \cup (\mathsf{dec}(\Phi_1) \oplus \mathsf{dec}(\Phi_2))$.

The components of a γ-formula θ are finally defined as follows.

Definition 8. *Let* $\theta = \langle\langle A\rangle\rangle \Phi$ *or* $[[A]]\Phi$ *be a* γ*-formula. All triples* $\langle \psi, \Psi, \Gamma\rangle$ *in* $\mathsf{dec}(\Phi)$ *are converted to a* γ*-set of formulae* $\gamma_s(\psi, \Psi, \Gamma) = \Gamma$*, and to a* γ*-component of* θ *that is a state formula* $\gamma_c(\psi, \Psi, \Gamma)$ *defined as follows:*

- $\gamma_c(\psi, \Psi, \Gamma) = \psi$ *if* Ψ *is* $\top$
- $\gamma_c(\psi, \Psi, \Gamma) = \psi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Phi$ *if* $\theta = \langle\langle A\rangle\rangle \Phi$
- $\gamma_c(\psi, \Psi, \Gamma) = \psi \wedge [[A]] \bigcirc [[A]]\Phi$ *if* $\theta = [[A]] \bigcirc [[A]]\Phi$ *if* $\theta = [[A]]\Phi$

As an example, consider again γ–formula $\langle\langle A\rangle\rangle((\Box p) \vee (\Box q))$.
Here $\mathsf{dec}((\Box p) \vee (\Box q))=$
$\mathsf{dec}(\Box p) \cup \mathsf{dec}(\Box q) \cup (\mathsf{dec}(\Box p) \oplus \mathsf{dec}(\Box q)) =$
$\{\langle p, \Box p, \{p\}\rangle, \langle q, \Box q, \{q\}\rangle, \langle p \wedge q, (\Box p) \vee (\Box q), \{(\Box p) \vee (\Box q)\}\rangle\}$.
We get three γ-components:
$p \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Box p$,
$q \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Box q$,
and $p \wedge q \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle (\Box p) \vee (\Box q)$.

A lemma in [Dav15a] states that each formula $\langle\langle A\rangle\rangle \Phi$ is equivalent to the disjunction of its γ components, and analogously for $[[A]]\Phi$.

Static Rule. Modulo these different notions of classification et decomposition of non-primitives formulae, therefore of full saturation of a consistent set of (state) formulae, the SR rule for this tableau calculus stays essentially the same as for the calculus already seen for LTL.

Let Γ be a set of state ATL* formulae; define Γ is *patently consistent* exactly as it is done for the case of LTL, and, again, let's just say that Γ is consistent when it is not patently inconsistent.

A *full expansion* of a consistent set of formulae Γ is a set of formulae Δ obtained by applying the following iterative procedure (that obviously halts):

- Initialisation: Set $\Delta := \Gamma$
- Saturation:
  - (α-Saturation). If $\varphi \in \Delta$ is an α-formula and it is not marked as "used", then add to Δ both its components and mark φ as "used";
  - (β-Saturation). If $\varphi \in \Delta$ is a β-formula and it is not marked as "used", then add to Δ one of its components and mark φ as "used".
  - (γ-Saturation). If $\varphi \in \Delta$ is a γ-formula and it is not marked as "used", then add to Δ one of its components and mark φ as "used".
    Moreover, if the added γ-component, is, say, $\gamma_c(\psi, \Psi, \Lambda)$ associate the set of path formulae $\gamma_s((\psi, \Psi, \Lambda)$, namely Λ, to this component.

Let us note $FS(\Gamma)$ the family of all the full saturations of Γ. As for LTL, the rule (SR) expands a vertex Γ that is consistent and contains at least a non-primitive formula by creating all the elements of $FS(\Gamma)$ as Γ's successors (avoiding duplications). Moreover Γ is connected to each of its successors via a $\Rightarrow$ edge (tableaux for ATL* are graphs having two sorts of edges).

A vertex created as the result of the application of the SR-rule is said to be a *tableau state*, and corresponds to an elements of $\mathbb{S}$ for a CGM candidate to be a model of the tableau root, while a vertex to which SR is applied is called a *tableau prestate*. The root can always be declared to be a prestate, but it is worthwhile observing that it will always be, trivially, also a state in the particular case where it contains only primitive formulae.

It is worthwhile observing that associating each γ-component of a γ formula φ to a set of path formulae (the third element of the triple used to generate this component), thereby pairing it with a tableau state, is a form of book-keeping of constraints that need to be taken into account when checking for eventuality fulfillment in the elimination phase.

Dynamic Rule

The only dynamic rule of the calculus is the $Next$ rule, whose aim is to build the successors of the states of the current tableau. Such successors will be pre-states that, once saturated via SR, will become in their turn states of the tableau (corresponding to elements of $\mathbb{S}$ for the CGMs that are candidate to be models of the root of the tableau).

The main ideas that underlie the formulation of such a rule are intuitive enough, but the formulation of the rule is rather technical. These ideas are:

– The agents $\mathbb{A}$ performing actions are those explicitly mentioned in the root formula (tight satisfiability).
– All primitive successor formulae of the state Δ to which $Next$ is applied are arranged in a list $\mathbf{L}$ where all the formulae of the form $\langle\langle A\rangle\rangle \bigcirc \varphi$ (existential successor formulae) precede all the formulae of the form $[[A]] \bigcirc \psi$ (universal successors formulae). Hence each element of $\mathbf{L}$ receives a number, corresponding to its position in the list. An action of an agent a is identified with the choice of the number of the formula that a decides to enforce. The case of formulae of the form $[[\mathbb{A}]]\mu$ (recall that $\mathbb{A}$ is the set of all the agents) is particular, because *all* the successors of Δ must satisfy the formula μ, thus a unique dummy action can be assigned to each agent to enforce μ.
– If there are k possible actions (k successor formulae in Δ), there will be $k^{|\mathbb{A}|}$ possible global actions that can be played at Δ. Each of them – let's note it $\sigma_{\mathbb{A}}$ – will connect Δ to a successor vertex; let's say that such a vertex is the target of $\sigma_{\mathbb{A}}$. Each $\sigma_{\mathbb{A}}$ can be seen as a program encoding which successor components will be in in its target, and can be seen as a "collective vote " made by all agents.
– This program (or "vote") must be such that the semantics of the strategy quantifiers is respected.
For each existential formula $\langle\langle A\rangle\rangle \bigcirc \varphi \in \mathbf{L}$ this means that $\sigma_{\mathbb{A}}$ must encode an A-action that guarantees φ to be true at each state that is issued via the saturation of $\sigma_{\mathbb{A}}$'s target, no matter how the other agents act. This is easy to achieve, by setting that each $\sigma_{\mathbb{A}}$ where all the agents in A have chosen (the number of) $\langle\langle A\rangle\rangle \bigcirc \varphi$ leads to a successor vertex containing φ.
The situation is more delicate for universal formulae $[[A]] \bigcirc \psi \in L$ where $A \neq \mathbb{A}$. Here, what is must assured is that for each A-action there is at least one "answer" of the opponent coalition, namely $\mathbb{A} \setminus A$, that enforces ψ. Hence the existence of such answers must be guaranteed. This is achieved by means of the function co defined below. Very roughly, the idea is the following. Let $[[A]] \bigcirc \psi \in \mathbf{L}$ be the i-th universal formula in $\mathbf{L}$. The function co is defined in such a way that each A-action σ_A can always be completed so to obtain at least one $\sigma_{\mathbb{A}}$ such that $\mathsf{co}(\sigma_{\mathbb{A}}) = i+1$: the agents in $\mathbb{A} \setminus A$ can synchronise to do so. The resulting global vector will lead to a successor containing ψ.

Formally, the $Next$ rule is defined as follows (abbreviating $\sigma_{\mathbb{A}}$ by σ):

Let Δ a tableau state, and let σ be a shorthand for $\sigma_{\mathbb{A}}$. Do:

1. List all primitive successor formulae in Δ in such a way that all existential formulae $\langle\langle A\rangle\rangle \bigcirc \varphi$ precede all the universal ones $[[A']] \bigcirc \varphi$, where $A' \neq \mathbb{A}$, and and put at the end of the list, all the formulae of the form $[[\mathbb{A}]] \bigcirc \varphi$; let the result be the list $\mathbf{L}$:

   $[\langle\langle A_0\rangle\rangle \bigcirc \varphi_0, ..., \langle\langle A_{m-1}\rangle\rangle \bigcirc \varphi_{m-1}, [[A'_0]] \bigcirc \psi_0, ..., [[A'_{l-1}]] \bigcirc \psi_{l-1}, [[\mathbb{A}]] \bigcirc \mu_0, ..., [[\mathbb{A}]] \bigcirc \mu_{n-1}]$
   Let $r_\Delta = \max(m + l, 1)$

Let $D(\Delta)$ be the set of action vectors $\{0, ..., r_\Delta - 1\}^{|\mathbb{A}|}$.
for each $\sigma \in D(\Delta)$, set :
• $N(\sigma) = \{i \mid \sigma_i \geq m\}$, where σ_i is the ith component of the tuple σ,
• $\mathsf{co}(\sigma) = [\Sigma_{i \in N(\sigma)}(\sigma_i - m)] \; mod \; l$.

2. For each $\sigma \in D(\Delta)$ create a prestate Γ_σ defined as:

$$\{\varphi_p \mid \langle\langle A_p \rangle\rangle \bigcirc \varphi_p \in \Delta \text{ and } \sigma_a = p \text{ for each agent } a \in A_p \}$$

$$\cup$$

$$\{\psi_q \mid [[A'_q]] \bigcirc \psi_q \in \Delta, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} \setminus A'_q \subseteq N(\sigma)\}$$

$$\cup$$

$$\{\mu_r \mid [[\mathbb{A}] \bigcirc \mu_r \in \Delta\}$$

If $\Gamma_\sigma = \emptyset$ then add $\top$ to it.
Then connect Δ to Γ_σ via the labelled edge $\xrightarrow{\sigma}$. If, however, $\Gamma_\sigma = \Gamma$ for some already existing prestate Γ, then do not create a second copy of Γ but only connect Δ to Γ via $\xrightarrow{\sigma}$.

For more details on the *Next* rule the interested reader may consult [Dav15a]; but it is rather in [Dav15b] that the intuitive meanings of all the technicalities of the rule formulation are fully explained.

Figure 8 shows the output of the construction phase for the formula

$$\langle\langle H \rangle\rangle(((\neg p \wedge \neg r)\mathsf{U}l) \wedge \Diamond r) \wedge \langle\langle B \rangle\rangle \Box \neg l$$

where H and B are agents and p, r and l are propositional variables. In the figure, vertices containing a patent contradiction $v, \neg v$ for some propositional letter v are not shown (they will get eliminated anyway in the elimination phase, see later on).

6.2 Elimination Phase

This phase bears similarities to the corresponding one for Wolper's tableaux for LTL, and proceeds by iteratively eliminating vertices of the current tableau that cannot contribute to models of the root until stability of the resulting tableau. In particular, as for LTL, vertices showing that some eventuality will never be fulfilled need to be eliminated. However, handling eventualities ATL* is more involved than in LTL. Two difficulties have to be faced.

The first one is: what exactly is to be considered an eventuality here? For instance, certainly $\langle\langle A \rangle\rangle p\mathsf{U}q$ is an eventuality"promising" that q will eventually occur. But what about the formula $\langle\langle A \rangle\rangle((\Box r) \vee p\mathsf{U}q)$? It is worthwhile recalling that this last γ-formula can *not* be analyzed as $(\langle\langle A \rangle\rangle \Box r) \vee \langle\langle A \rangle\rangle p\mathsf{U}q$. The chosen solution is to declare any formula containing an occurrence of U (necessarily positive because of the fnn) to be a *potential eventuality*.

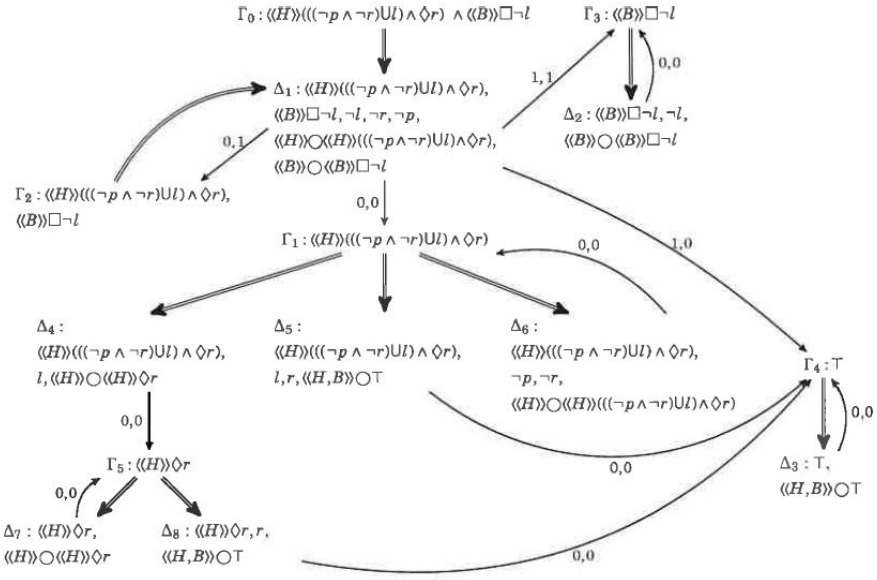

Fig. 8. A tableau for $\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U}l) \wedge \Diamond r) \wedge \langle\langle B\rangle\rangle\Box\neg l$ resulting from the construction phase

The second difficulty is caused by the fact that the "promised" formula might contain itself temporal operators, as, for instance, in the case of $\langle\langle A\rangle\rangle\Diamond\Box q$, where the promise is $\Box q$. Whenever the realization of $\Box q$ is initiated at a state s of the tableau by putting q in it, one must be memorize (via some sort of book-keeping) that *the present occurrence of q is just the beginning of the realization of $\Box q$ and that q must continue to be true on paths issued from s, in order to truly maintain the promise.* The role of the third element in the set of couples generated by Definitions 7 and 8 is exactly to make such a book-keeping possible.

For instance, a decomposition of $\langle\langle A\rangle\rangle\Diamond\Box q)$ (that is $\langle\langle A\rangle\rangle\top\mathsf{U}\Box q)$ produces two γ-components γ_{c1} and γ_{c1} and two corresponding γ-sets γ_{s1} and γ_{s2}:

1. $\gamma_{c1} = \top \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Diamond\Box q$ and $\gamma_{s1} = \{\top\}$
2. $\gamma_{c2} = q \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Diamond\Box q$ and $\gamma_{s2} = \{q, \Box q\}$

The second case corresponds to the situation where the promised formula $\Box q$ is initiated at a tableau state and a link from that state to $\gamma_{s2} = \{q, \Box q\}$ keeps track of the fact that q has started to be true but must continue to be true.

Based on these ideas a notion of *realization of a potential eventuality at a state Δ of a tableau $\mathcal{T}$* is defined. The definition is technical enough, and we do not recall it here; the interested reader may read [Dav15a]. Modulo this new definition of realization of eventualities, the elimination phase consists in iterating the following rules, until stability of the tableau is reached.

RULE ER1 .

1. If a tableau state Δ of a tableau $\mathcal{T}$ is patently inconsistent, then update $\mathcal{T}$ by erasing Δ.
2. If a tableau prestate Γ of a tableau $\mathcal{T}$ is such that each node Δ that is a successor of Γ via $\Rightarrow$ has been already eliminated, then update $\mathcal{T}$ by erasing Γ.
3. If a tableau state Δ of a tableau $\mathcal{T}$ is such that for some $\sigma \in D(\Delta)$[8] the node target of a $\xrightarrow{\sigma}$ arrow stemming from Δ has been eliminated, then update $\mathcal{T}$ by erasing Δ.

RULE ER2 If Δ is a tableau state containing a potential eventuality that is not realized at Δ in $\mathcal{T}$, then update $\mathcal{T}$ by erasing Δ.

We have not given here the technical definition of "realization of a potential eventuality φ at a tableau state Δ", that can be found in [Dav15a]; intuitively, it means that the fulfillment of a formula promised by φ is witnessed in the tableau.

Given a tableau $\mathcal{T}$ for a set of formulae Γ, $\mathcal{T}$ is declared *open* if the root Γ survives to the elimination phase, and closed otherwise.

Theorem 1. *The calculus is correct an complete with respect to unsatisfiability: a tableau for Γ is closed if and only if Γ is unsatisfiable [Dav15a]. The procedure terminates and it runs in at most 2EXPTIME, which is the intrinsic complexity of the satisfiability decision problem for* ATL*. [9]

7 Concluding Remarks

The examination of proofs respectively sequent-based and tableau-based shows that while the two approaches essentially coincide for (classical logic and) several modal logics, they have a more involved relation for temporal logics. We have illustrate this focussing on LTL, a particular temporal logic that is at the hearth of a lot of temporal logics widely used in computer science to model the dynamic

[8] See the meaning of the notation $D(\Delta)$ in the previous description of the *Next*-rule.

[9] Actually, the argumentation given in [Dav15a] concludes only to 3EXPTIME, but it has been refined in [Dav15b] to get the optimal complexity.

behavior of systems. We have also provided elements to explain why in the case of temporal logics the definition of "simple" sequent calculi, using only local conditions, and of related simple tableau calculi has to face some difficulties.

Then we have described with some details the tableau system proposed in [Dav15a] for ATL^*, allowing for the decision of the satisfiability/validity problems, under a tight notion of satisfiability and a perfect recall semantics. Such a tableau calculus has no sequent calculus counterpart, up to our knowledge. In fact, the only other known method to decide the ATL^* satisfiability problem is automata based [Sch08].

As a future work, we plan to examine the possibility of defining a sequent calculus for ATL^* based on the labelled approach described in [Neg05]. This approach, that allows for the expression of semantical relation between vertices of an interpretation via a specific kind of formulae, has revealed fruitful for Priorean's time logic ([BN09]). However that work has a computational drawback because its complexity is far from the optimal structural complexity of the satisfiability decision problem for that logic. As we observed, the satisfiability problem for ATL^* is already 2EXPTIME, so that one can hope that the intrinsic exponential nature of a similar approach to ATL^* would not be an hindrance to obtain an optimal complexity of the resulting sequent calculus.

References

[AHK97] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100–109, October 1997.

[AHK02] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[AJL19] Bahareh Afshari, Gerhard Jäger, and Graham E. Leigh. An infinitary treatment of full mu-calculus. In *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, pages 17–34, 2019.

[AL17] Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017.

[Bel01] Nuel Belnap. *Facing the Future: Agents and Choices in Our Indeterminist World*. Oxford University Press on Demand, 2001.

[Bet56] E. W. Beth. On padoa's method in the theory of definition. *Journal of Symbolic Logic*, 21(2):194–195, 1956.

[Bet59] E. W. Beth. Semantic entailment and formal derivability. *Sapientia*, 14(54):311, 1959.

[BL08] Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *J. Log. Algebr. Program.*, 76(2):216–225, 2008.

[BN09] Bianca Boretti and Sara Negri. Decidability for priorean linear time using a fixed-point labelled calculus. In *Automated Reasoning with Analytic Tableaux and Related Methods, 18th International Conference, TABLEAUX 2009, Oslo, Norway, July 6-10, 2009. Proceedings*, pages 108–122, 2009.

[Bor08] Bianca Boretti. *Proof Analysis in Temporal Logic, Ph.D. Thesis*. PhD thesis, University of Milan Italy, 2008.

[CDG14] S. Cerrito, A. David, and V. Goranko. Optimal tableau method for constructive satisfiability testing and model synthesis in the alternating-time temporal logic atl+. In *Proceedings of IJCAR 2014*, volume LNAI 8652. Springer, 2014.

[CDG15] Serenella Cerrito, Amélie David, and Valentin Goranko. Optimal tableau method for constructive satisfiability testing and model synthesis in the alternating-time temporal logic atl+. *ACM Trans. Comput. Logic*, 17(1):4:1–4:34, October 2015.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[Cur52] Haskell B. Curry. The elimination theorem when modality is present. *The Journal of Symbolic Logic*, 17(4):249–265, 1952.

[Dav15a] Amélie David. Deciding atl* satisfiability by tableaux. In *25th International Conference on Automated Deduction (CADE 2015)*, volume 9195 of *Lecture Notes in Computer Science*, pages 214–228, Berlin, Germany, August 2015.

[Dav15b] Amélie David. *Towards Synthesizing Open Systems: Tableaux For Multi-Agent Temporal Logics. (Vers la Synthèse de Systèmes Ouverts : Tableaux pour les Logiques Temporelles Multi-Agents).* PhD thesis, University of Paris-Saclay, France, 2015.

[DBHS16] Amina Doumane, David Baelde, Lucca Hirschi, and Alexis Saurin. Towards completeness via proof search in the linear time μ-calculus: The case of büchi inclusions. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 377–386, 2016.

[DGHP99] Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Springer, 1999.

[DGL16] Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016.

[DHL06] Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, pages 273–284, 2006.

[Fre79] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinenDenkens*. Verlag von Louis Nebert, Halle, 1879. siehe auch Wikipedia:Begriffsschrift.

[Gal15] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving, Second Edition*. Dover Publications, Inc., New York, NY, USA, 2015.

[GHL+09] Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual systems of tableaux and sequents for PLTL. *J. Log. Algebr. Program.*, 78(8):701–722, 2009.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, January 1987.

[GMR17] Nicola Gigante, Angelo Montanari, and Mark Reynolds. A one-pass tree-shaped tableau for ltl+past. In *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, pages 456–473, 2017.

[GvD06] V. Goranko and G. van Drimmelen. Complete axiomatization and decidablity of Alternating-time temporal logic. *Theor. Comp. Sci.*, 353:93–117, 2006.

[Hil22] David Hilbert. Neubegründung der mathematik. erste mitteilung. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 1(1):157–177, 1922.

[JB11] Wojciech Jamroga and Nils Bulling. Comparing variants of strategic ability. pages 252–257, 01 2011.

[JKS08] Gerhard Jäger, Mathis Kretz, and Thomas Studer. Canonical completeness of infinitary mu. *J. Log. Algebr. Program.*, 76(2):270–292, 2008.

[Kaw87] Hiroya Kawai. Sequential calculus for a first order infinitary temporal logic. *Math. Log. Q.*, 33(5):423–432, 1987.

[Kri63a] Saul A. Kripke. Semantical analysis of modal logic i. normal propositional calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 9(56):67–96, 1963.

[Kri63b] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16(1963):83–94, 1963.

[Neg05] Sara Negri. Proof analysis in modal logic. *J. Philosophical Logic*, 34(5-6):507–544, 2005.

[Nv16] Sara Negri and Jan von Plato. *Cut elimination in sequent calculi with implicit contraction, with a conjecture on the origin of Gentzen's altitude line construction*, pages 269–290. Number 6 in Ontos Mathematical Logic. de Gruyter, Germany, 2016.

[NvP14] Sara Negri and Jan von Plato. *Proof Analysis - A Contribution to Hilbert's Last Problem*. Cambridge University Press, 2014.

[OB03] Jens Otten and Wolfgang Bibel. leancop: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161, 2003.

[OM57] Masao Ohnishi and Kazuo Matsumoto. Gentzen method in modal calculi. *Osaka Mathematical Journal.*, (2):113–130, 9 1957.

[Pae88] Barbara Paech. Gentzen-systems for propositional temporal logics. In *CSL '88, 2nd Workshop on Computer Science Logic, Duisburg, Germany, October 3-7, 1988, Proceedings*, pages 240–253, 1988.

[Pel01] Doron A. Peled. *Software Reliability Methods*. Texts in Computer Science. Springer, 2001.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57, 1977.

[Pra71] D. Prawitz. Ideas and results in proof theory. 1971.

[Rey16] Mark Reynolds. A new rule for LTL tableaux. In *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016*, pages 287–301, 2016.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965.

[Sch98] Stefan Schwendimann. A new one-pass tableau calculus for PLTL. In *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '98, Oisterwijk, The Netherlands, May 5-8, 1998, Proceedings*, pages 277–292, 1998.

[Sch08] Sven Schewe. Atl* satisfiability is 2exptime-complete. *Automata, Languages and Programming*, pages 373–385, 2008.

[StL19] Special issue: General proof theory. *Stud Logica*, 107(1):1–5, 2019.

[Sza71] M. E. Szabo. The collected papers of gerhard gentzen. *Journal of Philosophy*, 68(8):238–265, 1971.

[Wol85] Pierre Wolper. The tableau method for temporal logic: An overview. *Logique Et Analyse*, 28(110-111):119–136, 1985.