Branching Path Planning with Modal Logics

Everardo Bárcenas^{1,2}, Edgard Benítez-Guerrero^{2}, Antonio Benitez^{3}, Jorge de la Calleja^{3}, Ma. Auxilio Medina^{3}

^{1} CONACYT, Mexico

^{2} Universidad Veracruzana, Mexico

^{3} Universidad Politécnica de Puebla, Mexico

iebarcenaspa@conacyt.mx, edbenitez@uv.mx, {antonio.benitez,jorge.delacalleja,maria.medina}@uppuebla.edu.mx

Abstract. Path planning concerns about finding a route for an agent, such that this agent move along the route from an initial to a final goal. Some additional constraints may also have to be satisfied for the agent, such as avoiding obstacles or collisions. Path planning has been recently studied in the context of linear temporal logic with great success. Expressive constraint specifications involving temporal ordering can be succinctly expressed by logic formulas, whereas environments are abstracted as transition systems. The plan is obtained by counterexample generation in a model checking tool: finding a path, if any, such that a given formula (constraints) satisfies a given model (agent environment). Due to the expressive power of linear temporal logic, only linear planning has mostly been considered so far, that is, plans corresponding to tasks to be performed in a linear successive order. In this work, we study branching shaped (tree) plans in the context of the μ-calculus, an expressive modal logic which subsumes many program logics such as LTL, PDL and CTL. Branching plans can be succinctly expressed by logic formulas so that a team of mobile devices can concurrently satisfy the plan. In the current work, we provide a plan generator based on a model checking algorithm for the μ-calculus. We show the algorithm is sound and complete, that is, for any environment, there a satisfying plan for a given set of constraints, if and only if, the plan generator succeeds.

Keywords. Branching shaped (tree) plans, μ-calculus, modal logic, plan generator, model checking.

1 Introduction

Path planning of agents (mobile devices including autonomous robots) consists in finding a sequence of motion tasks to reach a final goal in a known environment. Usually, constraints such as avoiding obstacles and collisions must also be satisfied. Constraints or specifications are usually given in a mathematical language, which traditionally has been a geometric one. It is then a natural research direction to study specification languages with more expressive power in order to give more complex instructions to agents.

Specifications involving sequentiality has recently been studied with great success in the setting of linear temporal logic [1, 23, 13, 14, 6, 25]. Specification are written as linear temporal logic formulas, and plan generation corresponds to counterexample generation in a model checking tool: given a formula (specification) and a model (environment), find a path in the model not satisfying (the counterexample) the negation of a formula.

In the context of mobile robotics, it is often required to implement continuous trajectories. This is achieved by first giving a discrete abstraction of the robot environment, which can clearly be obtained by well known efficient decomposition methods for polygonal environments [21]. A discrete plan is then obtained from the discrete abstraction of the environment and the specification by means of model checking techniques. A continuous implementation of the discrete plan can be obtained with a hybrid control approach [11].

As suggested by its name, linear temporal logic (LTL) formulas can express linear plans only. In linear plans, agents execute tasks in a linear sequential order. Consider for instance the following formula in linear temporal logic:

$$\diamond p \wedge \diamond (\diamond q)$$

This expression requires a agent to visit p after one step, and after two steps to visit q. Hence, in a linear model, the above expression implies q is after p. However, in many cases it may not be true, say for instance that q is in a different path than p. Then, the resulting plan has a two branch tree form. With these kind of non-linear plans, more precisely branching plans, it is then easy to see that a team of agents may concurrently execute the plan, one agent each branch. In addition, since plans are tree shaped, collision-free plans are guaranteed and agents may be executed asynchronously.

In the community of context-aware computing, it has recently emerged the notion of context-aware path planning [24], which concerns the path planning problem with particular context variables included in the set of constraints, as for instance, location, time, weather, and community sense.

For instance, for this last variable, a well-known scenario arises in the generation of evacuation plans. It is often the case that evacuation plans need to take in consideration a team of people (with an smart phone each or some). In this scenario, it is desirable, whenever possible, each (or at least some) people follow a different evacuation plan in order to avoid bottlenecks and expedite the evacuation. Branching tree planning in the evacuation scenario offers several distinct and collision-free evacuation plans.

In the current work, we study the μ-calculus, which is an expressive modal logic, in the context of branching path planning for mobile agents (devices, robots, etc.). The μ-calculus is one of the most expressive, yet computable, logic languages. The expressive power of the μ-calculus corresponds the bisimulation invariant fragment of monadic second order logic [5], and hence, by the Rabin's Tree Theorem [20], also corresponds to tree automata and regular languages (grammars).

The μ-calculus is also known as the queen of the program logics, because it subsumes the linear temporal logic LTL, the propositional dynamic logic PDL, the computational tree logic CTL, and many expressive descriptive logics [7].

1.1 Motivations and Related Work

Planning or scheduling is one of the first fundamental problems in the Artificial Intelligence research community. This problem concerns the automation of strategies or action sequences of intelligent agents or autonomous robots. In this context, formal models described by action languages has a long tradition which dates back to the McCarthy's situation calculus [18].

Although efficient reasoning algorithm associated to modal languages are now mature enough at the industrial level [8, 15], there has been surprisingly relatively little research on planning in the setting of modal logic [12]. The mobile robotics community is another story, very recently, there is a major increasing research interest on modal (temporal) languages as specification language for path planning [1, 23, 13, 14, 6, 25].

Since linear temporal logic is not expressive enough for denoting multi-branching, there are still few studies on modal languages in the context of multi-agent systems, particularly mobile robots [16, 23]. In [16], it is presented a multi-agent approach to motion planning with respect to a common global goal. Specifications are written in linear temporal logic. Hence, robots cooperate to achieve a common goal. However, specifications are still restricted to produce linear plans. Another issue is that this approach is not complete, that is, some times a solution is not found even if it exists. In [23], multi-agent motion planning is studied in the setting of optimal plans. Specifications are also described in terms of linear temporal logic formulas. The team of agents is controlled by a timed automaton. Also, the team of agents cooperate to achieve a common goal. In contrast with the approach reported in [16], agents can move asynchronously.

In this work, we take a different approach from works in [16, 23], instead of complementing the linear temporal logic formalism with a multi-robot solution, we extend the expressive power of temporal logic so that in a unifying framework, multi-agent and expressive specifications can be handled. Models in μ-calculus produce multi-branching plans that can be executed by a team of agents, one each branch.

In contrast with other branching modal logics as CTL, the μ-calculus subsumes LTL, which has been largely studied and benchmarked before in the context of planning [1, 23, 13, 14, 6, 25]. Moreover, due to the correspondence of the μ-calculus and regular languages [5, 20], many other complex specifications not available in LTL, CTL and PDL, as for instance regular (tree) expressions (well known in programming languages), can be succinctly denoted.

Previous studies of path planning in the setting of LTL [1, 23, 13, 14, 6, 25] are based in counterexample generation in a model checking tool. The model checking problem consists in deciding whether a given formula (specification) is satisfied by a given model (agent environments). Counterexample generation consists in finding a an example where the specification fails to be satisfied. Hence, in the setting of planning, a plan is obtained by a counterexample of the specification negated. However, often in model checking algorithms, counterexample generation is a blind spot in the corresponding tools [10]. In the current work, we propose a sequent based derivation system for the model checking problem of the μ-calculus. The planning algorithm, based on logical flow graphs, extracts non-linear (multi-branching) plans directly from the proofs produced by the derivation system.

1.2 Contributions and Outline

In Section 2 the propositional modal μ-calculus is introduced. Also, we introduce a formal definition of the problem of path planning in terms of logic formulas and Kripke structures (transition systems). Intuitively, Kripke structures are the discrete abstraction of agent environments. We also show in this Section there is plan for any satisfiable formula. This notion of planning is used in the correctness proof the path planner proposed in Section 3.

In a contribution of independent interest, we also introduce in Section 3 a sequent based derivation system for the model checking problem of the alternation-free fragment of the modal μ-calculus. The planning algorithm extracts, using logical flow graphs, branching plans directly from the proofs produced by the derivation system. It is also shown our proposal is sound and complete, that is, the planner returns a satisfying path, if and only if, there is a solution. We conclude in Section 4 with a discussion of the current work and a brief summary of further research perspectives.

2 A Modal Logic for Path Planning

In this Section, we present the propositional modal μ-calculus, which is an expressive modal logic equipped with least and greatest fixed-points. We also show how to use logic formulas as specifications for path planning. It is also formally defined the problem of path planning in terms of logic formulas and Kripke structures (graph models).

Through this paper, we consider countable sets only, hence, when written set, we denote countable set.

2.1 Syntax and Semantics

An alphabet or signature is a pair (P, X), such that

— P is a set of propositional variables; and

— X is a set of variables.

Unless otherwise stated, we consider a fixed alphabet.

Definition 1 (Syntax)**.** Given an alphabet, the set of μ-calculus formulas is given by the following grammar:

$$\phi := p \mid \neg\phi \mid \phi \vee \phi \mid \diamond\phi \mid x \mid \mu x.\phi$$

Formulas are interpreted as sets of nodes in Kripke structures (Definition 2). Intuitively, Kripke structures are graphs with labeled nodes, sometimes called transition systems. Propositions are true or false at each node, whereas negation and disjunction of formulas are interpreted as set complement and the union of sets, respectively. Modal formulas $\diamond\phi$ hold at nodes, such that there is at least one adjacent node where ϕ is true. $\mu x.\phi$ formulas are interpreted as least fixed-points. Intuitively, least fixed-points are used as operators for finite recursion.

Consider for instance formula $\diamond p_3$. In a graph model, this expression denotes nodes, such that there is at least one accessible node where p_3 is satisfied. Graphical representation of this example is depicted in Figure 1. $\diamond p_3$ holds in n_1 because n_3 is accessible from n_1 and p_3 is satisfied by n_3. Now consider the fixed-point formula $\mu x.p_1 \vee \diamond x$. This formula is true at nodes where p_1 holds or at nodes that can access, through finite recursive navigation on edges, to at least one node where p_1 is true. In Figure 1, $\mu x.p_1 \vee \diamond x$ holds at nodes $n_1, n_2, \ldots, n_k$. If we now consider formulas as specifications for mobile agents, then $\mu x.p_1 \vee \diamond x$ may be interpreted as a task for an agent, such that it must visit p_1 no matters how far p_1 is. Then the path traced by the formula can be seen as the path plan for the agent. However, it is natural to ask for another task, as for instance, also visit p_2 no matters how far it is, that is, $\mu x.p_2 \vee \diamond x$. This two requirements may be easily expressed by conjunction:

$$(\mu x.p_1 \vee \diamond x) \wedge (\mu x.p_2 \vee \diamond x)$$

But it may also be the case that p_2 is not on the path to p_1, or the other way around (see Figure 1), that is, the plan is not linear. These two tasks may be performed by a team of two agents. In Figure 1, this conjunction of tasks holds at n_1 only, but $\mu x.p_2 \vee \diamond x$ is satisfied by $n_1, n_3, \ldots, n_l$. This is the path to p_2 from n_1. Both paths, the one to p_1 and the one to p_2, form the non-linear plan required to do the tasks. Through this paper, we will

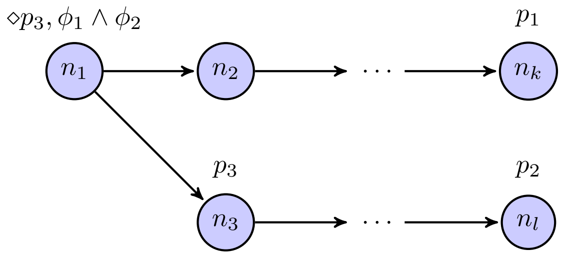


Fig. 1. The interpretation of $\diamond p_3$, $\phi_1 = \mu x.p_1 \vee \diamond x$, and $\phi_2 = \mu x.p_2 \vee \diamond x$

show how to compute non-linear path plans. First, we give a formal description of formula semantics on Kripke structures.

Given an alphabet (P, X), a Kripke structure is a tuple (N, R, L), such that

— N is a non-empty set of nodes;

— $R : N \times N$ is a binary relations of nodes; and

— $L : N \mapsto 2^P \setminus \emptyset$ is a total label function.

A total Kripke structure is a Kripke structure where each node is connected by at least one transition relation, that is, for every node $n \in N$ there is a relation R and a node $n' \in N$, such that $(n, n') \in R$. In this work, we consider total Kripke structures only.

Before giving an operational semantics of μ-calculus formulas, we need some notation. A valuation in a Kripke structure K is a function from the set of variables to the power set of nodes in K, that is, $V : X \mapsto 2^N$. We write $V\left[{}^S/_x\right]$, when $V(x) = S$.

Definition 2 (Semantics)**.** Given a Kripke structure $K = (N, R, L)$ and a valuation V, formulas are interpreted as follows.

$$\begin{aligned}
[\![p]\!]_V^K &= \{n \in N \mid p \in L(n)\}\\
[\![x]\!]_V^K &= V(x)\\
[\![\neg\phi]\!]_V^K &= N \setminus [\![\phi]\!]_V^K\\
[\![\phi \vee \psi]\!]_V^K &= [\![\phi]\!]_V^K \cup [\![\psi]\!]_V^K\\
[\![\diamond\phi]\!]_V^K &= \left\{n \in N \mid \exists n' \in N : R(n, n'), n' \in [\![\phi]\!]_V^K\right\}\\
[\![\mu x.\phi]\!]_V^K &= \bigcap \left\{S \subseteq N \,\middle|\, [\![\phi]\!]_{V\left[{}^S/_x\right]}^K \subseteq S\right\}
\end{aligned}$$

If $[\![\phi]\!]_V^K \neq \emptyset$, we say K satisfies ϕ. The model checking problem consists in automatically deciding whether or not a given Kripke structure satisfies a given formula. We say that a formula is satisfiable, if there is a Kripke structure satisfying the formula. In case there is no Kripke structure satisfying the formula, then it is said that the formula is unsatisfiable. When a formula is satisfied by any Kripke structure, the formula is valid.

Other traditional logic operators are defined as follows:

$$\begin{aligned}
\phi \wedge \psi &= \neg\left(\neg\phi \vee \neg\psi\right)\\
\bot &= p \wedge \neg p\\
\Box\phi &= \neg \diamond \neg\phi\\
\phi \,\mathsf{U}\, \psi &= \mu x.\left(\psi \vee [\phi \wedge \diamond x]\right)\\
\phi \,\mathsf{R}\, \psi &= \neg\left(\neg\phi \,\mathsf{U}\, \neg\psi\right)
\end{aligned}$$

Conjunction is interpreted as expected, as set intersection. $\bot$ is a contradiction. $\Box\phi$ holds at nodes where each adjacent node satisfies ϕ. Consider for instance formula $\Box p_2$, which denote nodes such that all its accessible nodes satisfies p_2.

In Figure 2, $\Box p_2$ does not hold at n_2, because although n_5 and n_4 are accessible nodes from n_2 and they satisfy p_2, there is one accessible node where p_2 is not satisfied and it is n_3.

A greatest fixed point can also be defined as dual of the least fixed point.

$$\nu x.\phi = \neg\mu x.\neg\phi\left[{}^{\neg x}/_x\right]$$

This operator is intuitively used for non-finite recursion. Consider for instance formula $\nu x.p_1 \wedge \diamond x$. This formula denotes paths (finite or infinite) where p_1 is always true. In the context of path planning, it is natural to require a task to be performed permanently, such as surveillance. In Figure 2, $\nu x.p_1 \wedge \diamond x$ denotes the cycle composed by $n_1, n_2, n_3, \ldots, n_k$.

Due to Knaster and Tarski we know μ and ν formulas are fixed points. Before describing this well known result, we need some notation.

Definition 3 (Finite expansions)**.** We define the finite expansion (approximants) of fixed-points as follows, for a natural number k.

$$\begin{aligned}
\mu^0 x.\phi &:= \phi\left[{}^{\bot}/_x\right] & \mu^k x.\phi &:= \phi\left[{}^{\mu^{k-1}x.\phi}/_x\right]\\
\nu^0 x.\phi &:= \phi\left[{}^{\top}/_x\right] & \nu^k x.\phi &:= \phi\left[{}^{\nu^{k-1}x.\phi}/_x\right]
\end{aligned}$$

We now recall the Knaster-Tarski Theorem in terms of finite expansions.

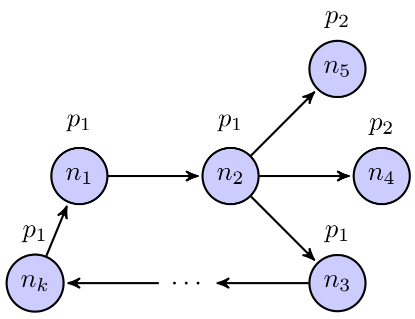


Fig. 2. Infinite recursion: the interpretation of $\nu x.p_1 \wedge \diamond x$

Theorem 1 (Knaster-Tarski Fixed-Point Theorem [22, 17]). *For any Kripke structure $K = (N, R, L)$ and a valuation V,*

$$[\![\mu x.\phi]\!]_V^K = \left[\!\!\left[\bigvee_{i=0}^{|N|} \mu^i x.\phi\right]\!\!\right]_V^K$$
$$[\![\nu x.\phi]\!]_V^K = \left[\!\!\left[\bigwedge_{i=0}^{|N|} \nu^i x.\phi\right]\!\!\right]_V^K$$

Other known operators in LTL and CTL can also be easily defined.

$$\phi \mathsf{U} \psi = \mu x. (\psi \vee [\phi \wedge \diamond x])$$
$$\phi \mathsf{R} \psi = \neg (\neg\phi \mathsf{U} \neg\psi)$$

The until formula $\phi \mathsf{U} \psi$ holds in each node of paths (adjacent nodes) where ϕ is true in all but he last node, where ϕ is satisfied. Release formula $\phi \mathsf{R} \psi$ holds in nodes of paths where ψ is true in all but the last node, where ϕ is true. Also, release formula holds in each node of non-finite paths where ψ is always true.

2.2 Plans

Path plans for mobile agents are defined as the resulting trees of nodes satisfying a formula and its subformulas in a given Kripke structure. Consider for example the following formula:

$$p_0 \wedge \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$$

This expression denotes a plan with two sub-tasks, the first one consists in, starting from p_0, visiting p_1 and p_2, in that order, whereas the second task also starts from p_0, but requires to visit p_3 and p_4, also in that order. In Figure 3, the Kripke structure satisfies the formula, the tree is the plan extracted from the Kripke structure that satisfies the formula together with its subformulas.

As another example, consider now formula $\nu x.p_1 \wedge \diamond x$ and its model displayed in Figure 2. The plan of the formula is then the one branched infinite tree composed by the infinite occurrence of paths composed by $n_1, n_2, n_3, \ldots, n_k$.

That a satisfiable formula is satisfied by a tree shaped Kripke structure is a well known result.

Theorem 2 (Tree model property [7]). *If a formula ϕ is satisfiable, that is, there is a Kripke structure K, such that for any valuation V we have that $[\![\phi]\!]_V^K \neq \emptyset$, then ϕ is satisfied by a tree shaped Kripke structure K', that is, $[\![\phi]\!]_V^{K'} \neq \emptyset$.*

Before defining a path plan in terms of Kripke structures and formulas, we need some notation.

Definition 4. A tree on a Kripke structure $K = (N, R, L)$, written T^K, is inductively defined as follows:

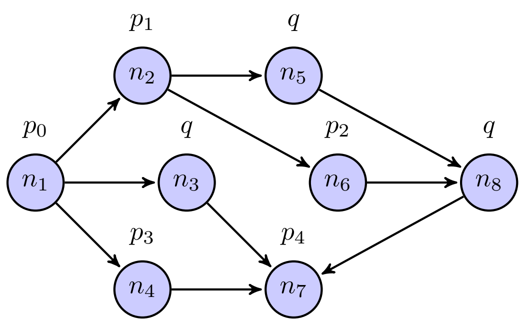
— the empty tree $\emptyset$ is a tree; and
— the tuple $(n, T_1, T_2, \ldots, T_k)$ is a tree, such that
  - $n \in N$ is a node called the root,
  - for $i = 1, \ldots, k$, T_i are trees, and
  - in case T_i is not empty, then $R(n, n_i)$, where n_i is the roots of T_i, respectively.

When clear from context, we often simply write T for a tree.

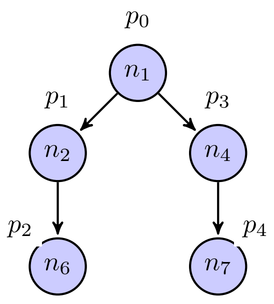
We now define the notion of plan. Intuitively, a plan is a tree whose nodes satisfy, following the paths of the tree, a formula together with its subformulas.

Definition 5 (Plan). Given a Kripke structure $K = (N, R, L)$ and a valuation V, we say a tree T^K on K is a plan for a formula ϕ if and only if $T^K \models \phi$, such that relation $\models$ is inductively defined as follows.

$$(n, T_1, \ldots, T_k) \models p \text{ if and only if } p \in L(n)$$
$$T \models \neg\phi \text{ if and only if } T \not\models \phi$$
$$T \models \phi \vee \psi \text{ if and only if } T \models \phi \text{ or } T \models \psi$$
$$(n, T_1, \ldots, T_k) \models \diamond\phi \text{ if and only if } n \in [\![\diamond\phi]\!]_V^K, \quad \exists n_{i \in \{1,\ldots,k\}} \in N : R(n, n_i), \quad n_i \text{ is the root of } T_i, \text{ and } T_i \models \phi$$
$$T \models \mu x.\phi \text{ if and only if } T \models \bigvee_{i=0}^{|N|} \mu^i x.\phi$$



(a) Kripke structure



(b) Plan

Fig. 3. A Kripke structure satisfying $p_0 \wedge \diamond(p_1 \wedge \diamond p_2) \wedge \diamond(p_3 \wedge \diamond p_4)$, together with its corresponding plan

Relation $\not\models$ is defined as follows.

$$(n, T_1, \ldots, T_k) \not\models p \text{ if and only if } p \notin L(n)$$
$$T \not\models \neg\phi \text{ if and only if } T \models \phi$$
$$T \not\models \phi \vee \psi \text{ if and only if } T \not\models \phi \text{ and } T \not\models \psi$$
$$(n, T_1, \ldots, T_k) \not\models \diamond\phi \text{ if and only if } n \in [\![\diamond\phi]\!]^K_V, \quad \forall n_{i=1,\ldots,k} \in N : R(n, n_i) \quad n_i \text{ is the root of } T_i, \text{ and } T_i \not\models \phi$$
$$T \not\models \mu x.\phi \text{ if and only if } T \not\models \bigvee_{i=0}^{|N|} \mu^i x.\phi$$

Because of the tree model property of μ-calculus (Theorem 2), we know that if there is model satisfying a formula, then there is also a plan (tree) satisfying the formula.

Corollary 1. *If a formula ϕ is satisfiable, that is, there is a Kripke structure K, such that for any valuation V we have that $[\![\phi]\!]^K_V \neq \emptyset$, then there is a plan T^K for ϕ, that is, $T^K \models \phi$.*

3 Path Planning

In this Section, we introduce a sequent-like system for the model checking problem, that is, given a formula and a Kripke structure, decide whether or not the Kripke structure satisfies the formula. Plans can be extracted from the proofs produced by the sequent system. It is also proven that the system is sound and complete.

The sequent system considered in this Section works with fixed-point alternation-free formulas in negation normal form. Occurrences of greatest and least fixed-points cannot then alternate. The negation normal form of a formula is defined as follows.

$$\text{nnf}(p) = \neg p \qquad \text{nnf}(x) = \neg x$$
$$\text{nnf}(\phi \vee \psi) = \text{nnf}(\phi) \wedge \text{nnf}(\psi) \qquad \text{nnf}(\diamond\phi) = \Box\text{nnf}(\phi)$$
$$\text{nnf}(\mu x.\phi) = \nu x.\text{nnf}(\phi)\,[^x/_{\neg x}]$$

For the rest of the paper, when we refer to the negation of a formula $\neg\phi$, we mean its negation normal form nnf(ϕ).

The set of subformulas of a formula is inductively defined as follows: if ϕ is a proposition, a variable or constant $\top$, then $subformula(\phi) = \{\phi\}$; if ϕ is a conjunction $\psi \wedge \psi$ or a disjunction $\psi \vee \varphi$, then $subformula(\phi) = \{\phi\} \cup subformula(\psi) \cup subformula(\varphi)$; and if ϕ is a modal formula $\diamond\psi$ or $\Box\psi$, or a negation $\neg\psi$, or a fixed point formula $\mu x.\psi$ or $\nu x.\psi$, then $subformula(\phi) = \{\phi\} \cup subformula(\psi)$. The set of formulas in a modality is inductively defined as follows: $mod(p) = mod(x) = mod(top) = \{\}$; $mod(\diamond\phi) = mod(\Box\phi) = subformulas(\phi)$; $mod(\phi \vee \psi) = mod(\phi \wedge \psi) = mod(\phi) \cup mod(\psi)$; $mod(\neg\phi) = mod(\mu x.\phi) = mod(\nu(.\phi)) = mod(\phi)$.

We say a formula ϕ occurs under the scope of a modality in another formula ψ, whenever $\phi \in mod(\psi)$. The set of free variables in a formula is inductively defined as follows: $free(x) = \{x\}$; $free(\top) = free(p) = \{\}$; $free(\neg\phi) = free(\diamond\phi) = free(\Box\phi) = free(\phi)$; $free(\phi \vee \psi) = free(\phi \wedge \psi) = free(\phi) \cup free(\psi)$; and $free(\mu x.\phi) = free(\nu x.\phi) = free(\phi) \setminus \{x\}$, provided x occurs in ϕ. We say a variable x occurs free in a formula ϕ, whenever $x \in free(\phi)$. Without loss of generality, we consider formulas where variables can only occur under the scope of a modality, and not free [5].

Sequents are defined as pairs, written $n \vdash \Delta$, where n is a node and Δ is a (possibly empty) set of formulas. Intuitively, node n syntactically satisfies each formula in

Δ. We say a sequent is empty if its set of formulas is empty.

Rules are pairs (C, A), where C is a set of sequents, called consequent, and A is a family of sequent sets, called premises. Rules are usually written as follows:

$$\frac{A}{C}$$

Intuitively, from premises A, we derive C. If A contains only empty sequents, then we say A is empty and there are no premises. If a rule has no premises, then we say this rule is an axiom. A derivation system is composed by a set of rules.

Definition 6 (Planning derivation system)**.** Given a Kripke structure $K = (N, R, L)$, we define the following rules with respect to a given formula.

$$\frac{p \in L(n); n \vdash \Delta; \Lambda}{n \vdash p, \Delta; \Lambda} \qquad \frac{p \notin L(n); n \vdash \Delta; \Lambda}{n \vdash \neg p, \Delta; \Lambda}$$

$$\frac{n \vdash \phi, \Delta; \Lambda \quad n \vdash \psi, \Delta; \Lambda}{n \vdash \phi \vee \psi, \Delta; \Lambda} \qquad \frac{n \vdash \phi, \psi, \Delta; \Lambda}{n \vdash \phi \wedge \psi, \Delta; \Lambda}$$

$$\frac{\exists n' : R(n, n'); n' \vdash \phi; n \vdash \Delta; \Lambda}{n \vdash \diamond\phi, \Delta; \Lambda}$$

$$\frac{\forall n' : R(n, n'); n' \vdash \phi; n \vdash \Delta; \Lambda}{n \vdash \Box\phi, \Delta; \Lambda}$$

$$\frac{n \vdash \bigvee_{i=0}^{|N|} \mu^i x.\phi, \Delta; \Lambda}{n \vdash \mu x.\phi, \Delta; \Lambda} \qquad \frac{n \vdash \bigwedge_{i=0}^{|N|} \nu^i x.\phi, \Delta; \Lambda}{n \vdash \nu x.\phi, \Delta; \Lambda}$$

Please note that for readability we use commas for the union of formulas in sequents, and for the union of sequents we use semi-colons. That is, for a formula ϕ and a set of formulas Δ, we write ϕ, Δ instead of $\{\phi\} \cup \Delta$, and for a sequent $n \vdash \Delta$ and a set of sequents Λ, we write $n \vdash \Delta; \Lambda$ instead of $\{n \vdash \Delta\} \cup \Lambda$.

Before defining derivation trees formally, consider the following formula evaluated in the Kripke structure displayed in Figure 3:

$$p_0 \wedge \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$$

The corresponding derivation tree is the depicted in Figure 4. Inference goes bottom-up, so the formula in the bottom is concluded from the axiom (empty sequents). We thus read that $n_0 \vdash p_0 \wedge \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$ is concluded from $n_0 \vdash p_0, \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$ by applying the inference rule corresponding to conjunction of formulas. This last expression is concluded from $n_0 \vdash \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$ by applying the inference rule corresponding to propositions.

$$\begin{array}{c}
n_6 \vdash; n_2 \vdash; n_7 \vdash; n_4 \vdash; n_0 \vdash \\ \hline
n_6 \vdash; n_2 \vdash; n_7 \vdash p_4; n_4 \vdash; n_0 \vdash \\ \hline
n_6 \vdash; n_2 \vdash; n_4 \vdash \diamond p_4; n_0 \vdash \\ \hline
n_6 \vdash; n_2 \vdash; n_4 \vdash p_3, \diamond p_4; n_0 \vdash \\ \hline
n_6 \vdash; n_2 \vdash; n_4 \vdash (p_3 \wedge \diamond p_4); n_0 \vdash \\ \hline
n_6 \vdash; n_2 \vdash; n_0 \vdash \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_6 \vdash p_2; n_2 \vdash; n_0 \vdash \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_2 \vdash \diamond p_2; n_0 \vdash \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_2 \vdash p_1, \diamond p_2; n_0 \vdash \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_2 \vdash p_1 \wedge \diamond p_2; n_0 \vdash \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_0 \vdash \diamond (p_1 \wedge \diamond p_2), \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_0 \vdash \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_0 \vdash p_0, \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4) \\ \hline
n_0 \vdash p_0 \wedge \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)
\end{array}$$

Fig. 4. A proof tree for $p_0 \wedge \diamond (p_1 \wedge \diamond p_2) \wedge \diamond (p_3 \wedge \diamond p_4)$ in the Kripke structure of Figure 3

In this case, we know that $p_0 \in L(n_0)$. It is now applied again the rule for conjunction, so for the premise we obtain $n_0 \vdash \diamond (p_1 \wedge \diamond p_2), \diamond (p_3 \wedge \diamond p_4)$ Then, we apply the rule for diamond formulas, so the premise is $n_2 \vdash p_1 \wedge \diamond p_2;\ n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$. This is because we know that $R(n_0, n_2)$.

Using conjunction rule, then the premise is now $n_2 \vdash p_1, \diamond p_2;\ n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$, which is concluded from $n_2 \vdash \diamond p_2;\ n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$ due to the proposition rule, in particular, because $p_1 \in L(n_2)$.

Using again the diamond rule, the premise is then $n_6 \vdash p_2;\ n_2 \vdash;\ n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$, because $R(n_2, n_6)$. For the next step we obtain the axiom $n_6 \vdash;\ n_2 \vdash;\ n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$, because of proposition rule, more precisely, because $p_2 \in L(n_6)$.

Analogously as just described above, we now apply successively the diamond, the conjunction and the diamond rule over $n_0 \vdash \diamond (p_3 \wedge \diamond p_4)$ to finally obtain as premise $n_6 \vdash;\ n_2 \vdash;\ n_7 \vdash;\ n_4 \vdash;\ n_0 \vdash$, which turns to be an axiom.

As another example, consider the corresponding proof tree in Figure 5 for $\nu x.p_1 \wedge \diamond x$ in the Kripke structure in Figure 2.

In order to formally define derivation trees, we will consider trees of sequents, that is, trees with sequents instead of nodes. Formally, given a sequent Λ and a derivation system, a derivation tree of Λ is a tree $(\Lambda, T_1, T_2, \ldots, T_k)$, such that

— $T_1, \ldots, T_k$ are all derivation trees of their respective sequent roots $\Lambda_1, \ldots, \Lambda_k$, and

— $\frac{\Lambda_1\ \Lambda_2\ \ldots\ \Lambda_k}{\Lambda}$ is an instance of a rule in the derivation system.

$$\begin{array}{c}
n_k \vdash; \ldots; n_2 \vdash; n_1 \vdash \\ \hline
\vdots \\ \hline
n_k \vdash \nu^1 x.p_1 \wedge \diamond x; \ldots; n_2 \vdash; n_1 \vdash \\ \hline
\vdots \\ \hline
n_3 \vdash; n_2 \vdash; n_1 \vdash \nu^4 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_3 \vdash \top; n_2 \vdash; n_1 \vdash \nu^4 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash \diamond\top; n_1 \vdash \nu^4 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash p_1, \diamond\top; n_1 \vdash \nu^4 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash p_1 \wedge \diamond\top; n_1 \vdash \nu^4 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash p_1 \wedge \diamond\top; n_1 \vdash p_1, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash; n_1 \vdash p_1, \diamond(p_1 \wedge \diamond\top), \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash; n_1 \vdash \nu^1 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_2 \vdash \top; n_1 \vdash \nu^1 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_1 \vdash \diamond\top, \nu^1 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_1 \vdash p_1, \diamond\top, \nu^1 x.p_1 \wedge \diamond x, \ldots, \nu^k x.p_1 \wedge \diamond x \\ \hline
n_1 \vdash (\nu^0 x.p_1 \wedge \diamond x) \wedge \ldots \wedge (\nu^k x.p_1 \wedge \diamond x) \\ \hline
n_1 \vdash \nu x.p_1 \wedge \diamond x
\end{array}$$

Fig. 5. A proof tree for $\nu x.p_1 \wedge \diamond x$ in the Kripke structure in Figure 2

We are now ready to formally define derivation trees satisfying a sequent for a given Kripke structure. Proof trees are derivation trees for a given sequent, such that a given Kripke structure satisfies the formula, intuitively, a proof tree containing finite branches with at least one ending with axioms.

Definition 7 (Proof tree)**.** Consider the planning derivation system, a proof tree of a sequent Λ is a derivation tree of Λ, such that there is at least one branch ending with only empty sequents.

We are now ready to state the main result of this work, which is that the derivation system is sound and complete. This means that whenever there is a Kripke structure satisfying a formula, the derivation system obtains a proof tree.

Theorem 3 (Model Checking)**.** *A formula ϕ is satisfiable by a Kripke structure K, if and only if, there is a proof tree of $n \vdash \phi$, such that n is a node in K.*

To prove the sequent system is correct, we need to show both directions of the double implication.

Theorem 4 (Completeness)**.** *If a formula ϕ is satisfiable by a Kripke structure K, then there is a node n in K, such that there is a proof tree of $n \vdash \phi$.*

Proof. The proof goes by structural induction on the input formula.

First consider the input formula is a proposition p. We then know $p \in L(n)$. Hence, $n \vdash p$.

For the case of negation, recall formulas are negated normal form, hence, the scope of the negation symbol can only involve a proposition, say p. Since we know n satisfies $\neg p$, then $p \notin L(n)$, then $n \vdash p$.

In case the input formula is a disjunction with form $\psi \vee \varphi$, we know at least one of ψ or φ is satisfied by n. Without loss of generality, assume ψ is satisfied by n. Then by induction we know $n \vdash \psi$, and hence $n \vdash \psi \vee \varphi$.

The case when the input formula is a conjunction is proven analogously as the case for disjunction.

Assume now the formula has the form $\diamond\psi$. Since n satisfies $\diamond\psi$, then there is at least one node n' satisfying ψ, such that $R(n, n')$. By induction we obtain $n' \vdash \psi$ and hence $n \vdash \diamond\psi$.

The for modal formulas is proven analogously as the case of diamond formulas $\diamond\psi$.

We consider now the case when a least fixed-point $\mu x.\psi$ is satisfied by n. By the Knaster-Tarski Fixed-Point Theorem 1, we know $\mu x.\psi$ is equivalent to its finite expansion $\bigvee_{i=0}^{k} \mu^i x.\psi$, where k is the number of nodes in K (Definition 3). We then obtain n satisfies the finite expansion. Hence, there is some i, such that n satisfies $\mu^i x.\psi$. A second induction on the structure of $\mu^i x.\psi$ is applied in order to obtain $n \vdash \mu^i x.\psi$. These cases are identical as the ones already proven above. We then obtain $n \vdash \bigvee_{i=0}^{k} \mu^i x.\psi$, hence $n \vdash \mu x.\psi$.

Finally, for the case of greatest fixed points we proceed analogously as for least fixed points. By the Knaster-Tarski Fixed-Point Theorem 1, we know $\nu x.\psi$ is equivalent to its finite expansion $\bigwedge_{i=0}^{k} \nu^i x.\psi$, where k is the number of nodes in K (Definition 3). In order to show there is a proof tree of the finite expansion we proceed by induction on i (the number of conjuncts). For each step of the induction on i, we also apply another induction on the structure of ψ. By assumption, we know n satisfies $\nu^0 x.\psi$, the base cases (when ψ is a proposition or a negated proposition) are trivial. Consider now ψ is of the form $\langle m \rangle \varphi$, then there is a m-connected node n' to n satisfying ψ, by induction we obtain $n' \vdash \psi$. $n \vdash \langle m \rangle \psi$ follows immediately. Since variables only occur under the scope of a modality and not free, and the fixed-points cannot alternate, other cases go smoothly by structural induction. We now consider the inductive step on i. In order to show $n \vdash \nu^i x.\psi$, we again proceed by structural induction on ψ. Base cases are immediate. If ψ is $\langle m\rangle\,\varphi$, it is not hard to see there is a m-connected node n' to n, such that $n' \vdash \varphi\left[{}^{\nu^{i-1}x.\varphi}/_{x}\right]$, which clearly implies $n \vdash \langle m\rangle\,\varphi\left[{}^{\nu^{i-1}x.\varphi}/_{x}\right]$. $n' \vdash \varphi\left[{}^{\nu^{i-1}x.\varphi}/_{x}\right]$ is obtained by induction on φ. In the base case, $n' \vdash x\left[{}^{\nu^{i-1}x.\varphi}/_{x}\right]$ follows from $n' \vdash \nu^{i-1}x.\varphi$, which is known by induction. Other inductive cases goes straightforward by considering all possible occurrences of x. We then conclude $n \vdash \bigwedge_{i=0}^{k} \nu^i x.\psi$. □

We now show the converse, that is, if we have a proof tree, then the formula is satisfiable by the Kripke structure.

Theorem 5 (Soundness)**.** *Given a Kripke structure K and a formula ϕ, if there is a proof tree of $n \vdash \phi$ for some node n in K, then $n \in [\![\phi]\!]_V^K$ for any valuation V.*

Proof. We proceed in this proof also by structural induction on the input formula.

If the formula is a proposition p, we then know $p \in L(n)$, then n satisfies p.

When the formula is a negation, then it has the form $\neg p$ (recall formulas are in negated normal form). We then know $p \notin L(n)$, and then n does not satisfy p, hence n satisfies $\neg p$.

Consider the case of $n \vdash \psi \vee \varphi$. The proof tree has then the following form:

$$\frac{\frac{T_1}{n\vdash\psi} \quad \frac{T_2}{n\vdash\varphi}}{n \vdash \psi \vee \varphi}$$

By induction we know that $n \in [\![\psi]\!]_V^K$ or $n \in [\![\psi]\!]_V^K$ for any V, then $n \in [\![\psi \vee \varphi]\!]_V^K$.

Conjunction is proven analogously as disjunction.

If the input formula is $\Box\psi$, then the proof has the form

$$\frac{\frac{T_1}{n_1\vdash\psi} \quad \frac{T_2}{n_2\vdash\psi} \quad \cdots \quad \frac{T_k}{n_k\vdash\psi}}{n \vdash \Box\psi}$$

for all $n_1, \ldots, n_k \in N$, such that $R(n, n_i)$. Then by induction, $n_{i=1,\ldots,k} \in [\![\phi]\!]_V^K$ for any V, and hence $n \in [\![\Box\psi]\!]_V^K$.

When the input is a diamond formula we proceed analogously as with box formulas.

Now, consider the case of $\mu x.\psi$. The proof has the following form:

$$\frac{\frac{\frac{T}{n\vdash\mu^0 x.\psi \ \ \ldots \ n\vdash\mu^k x.\psi}}{n\vdash\bigvee_{i=0}^{k}\mu^i x.\psi}}{n\vdash\mu x.\psi}$$

for natural number k, which is the cardinality of set of nodes in K. We know there is some i, such that $n \vdash \mu^i x.\psi$. However, we cannot use direct induction to show that $n \in [\![\mu^i x.\psi]\!]_V^K$ for any valuation V, because $\mu^i x.\phi$ is not a subformula of $\mu x.\phi$. We then proceed to use a second induction, now on the structure of $\mu^k x.\phi$. These cases are proven as the ones already proven above. Finally, we obtain $n \in [\![\mu x.\phi]\!]_V^K$.

The case for greatest fixed-points is proven in an analogous manner as the case for least fixed points. We know by assumption

$$\frac{\frac{\frac{T}{n\vdash\nu^0 x.\psi,\ldots,\mu^k x.\psi}}{n\vdash\bigwedge_{i=0}^{k}\nu^i x.\psi}}{n\vdash\nu x.\psi}$$

for natural number k, which is the cardinality of set of nodes in K. We proceed by a second induction, now on the structure of $\nu^i x.\phi$, in order to show $n \in [\![\nu^i x.\psi]\!]_V^K$ for all i. These cases are proven as the ones already proven above (recall fixed points cannot alternate). We then conclude $n \in [\![\nu x.\phi]\!]_V^K$. □

Since nodes are syntactically represented in proof trees, and node transitions are denoted by derivation rules corresponding to modal formulas (diamond and box), it is not hard to see that plans can be extracted from proof trees. Consider for instance the proof tree in Figure 4. The nodes involved in the plan are n_0, n_2, n_6, n_4 and n_7. Transitions are denoted by the application of rules involving modal subformulas, hence we obtain that $R(n_0, n_2)$, and $R(n_2, n_6)$, in that order, and $R(n_0, n_4)$ and $R(n_4, n_7)$.

It is then easy to see that the plan is a two-branch tree composed by n_0, n_2, and n_6, and n_0, n_4, and n_7. A graphical representation of the branching plan over the proof tree of Figure 4 is depicted in Figure 6, where the bold transitions (diamond rules) defines the topology of the branching plan. In case there are not finite branches to extract a plan, we then extract the plan from the branch producing a ν-cycle. For instance, in Figure 5, from the branch producing the cycle is easy to extract the plan, which involves $n_1, n_2, \ldots, n_k$, in that order.

In order to formalize the plan extraction process from proof trees, inspired from the notion of logical flow graphs [9], we now define a similar notion for nodes occurring in proof trees. From this logical graphs, we are then going to be able to directly obtain plans in the name of tree paths.

Definition 8 (Logical graphs)**.** Given a proof tree of a certain formula, we define a logical graph as a tuple (N, E), such that

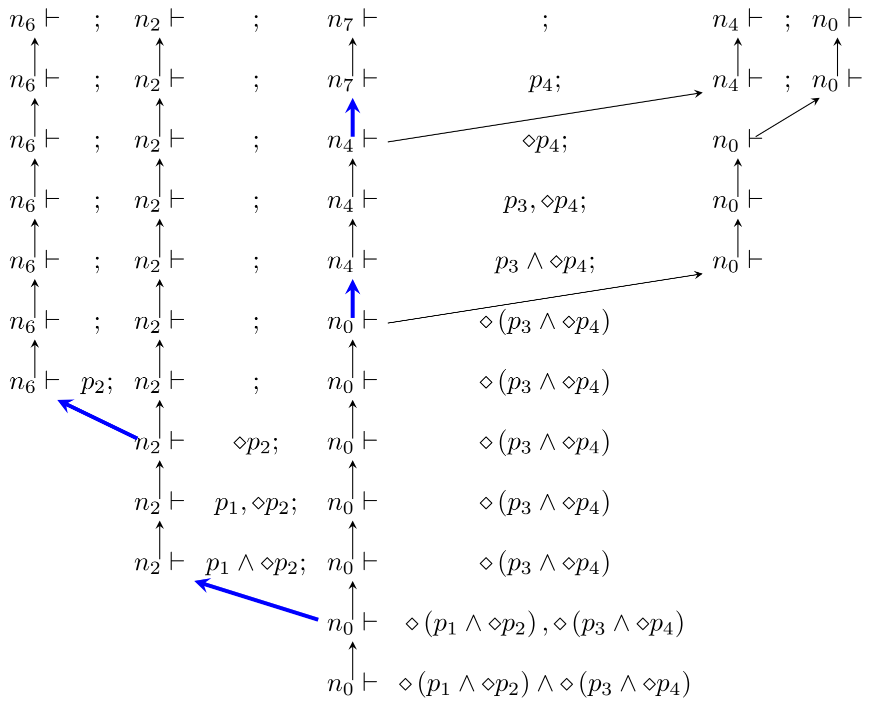


Fig. 6. A logical flow graph for $\diamond(p_1 \wedge \diamond p_2) \wedge \diamond(p_3 \wedge \diamond p_4)$: Kripke structure of Figure 3, proof tree of Figure 4

— N is a set containing the nodes occurring in the proof tree, and

— E is a binary relation $N \times N$, such that for each derivation step in the proof tree, if the step has the form:

- $\frac{n \vdash \psi, \Delta; \Lambda}{n \vdash \phi, \Delta; \Lambda}$, then $(n, n) \in E$; and
- $\frac{n' \vdash \phi, \Delta; \Lambda}{n \vdash \diamond\phi, \Delta; \Lambda}$ or $\frac{n' \vdash \phi, \Delta; \Lambda}{n \vdash \Box\phi, \Delta; \Lambda}$, then $(n, n') \in E$.

The logical graph of the proof tree displayed in Figure 3 is shown in Figure 6.

A tree path is simply the resulting tree of a logical graph without considering repetitions.

Definition 9 (Tree path)**.** Given a logical graph (N, E) of a proof tree of a formula ϕ, such that $n \vdash \phi$ is the root of the proof tree, we inductively define a tree path ϕ starting from n as follows:

— the root is n;

— if a node n' is in the tree path, $(n', n'') \in E$, and $n' \neq n''$, then n' is the parent of n''.

The tree path corresponding to the logical graph of proof tree in Figure 3 is graphically represented by the bold transitions in Figure 6.

Now, it is easy to imply from Theorem 3 that tree paths (Definition 9) are also plans (Definition 5).

Theorem 6 (Path planning)**.** *If there is a proof tree of a formula ϕ, then a tree path of ϕ is also a plan.*

Proof. The proof goes by induction on the structure of the input formula ϕ.

If the input formula is a proposition p, then we know there is a node n, such that $n \vdash p$. Hence, $p \in L(n)$, and then $n \models p$.

Negation, since it only occurs in front of propositions, is proven analogously as the case of propositions.

Consider now the input formula is a disjunction $\psi \vee \varphi$. We then know that there is a proof tree for either $n \vdash \psi$ or $n \vdash \varphi$. Without loss of generality assume there is a proof tree for $n \vdash \psi$. By induction we then obtain there is plan $(n, T_1, \ldots, T_k)$ for ψ, that is, $(n, T_1, \ldots, T_k) \models \psi$. By definition of plan (Definition 5), then $(n, T_1, \ldots, T_k) \models \psi \vee \varphi$.

The proof for the case of conjunctions is analogous as the proof for the case of disjunctions.

If the formula in question is $\diamond\psi$, and we know $n \vdash \diamond\psi$, then there is a node $n' \vdash \psi$, such that $R(n, n')$. By induction, it is known there is a plan $T' = (n', T_1, \dots, T_k) \models \psi$. Then, by Definition 5, we obtain a plan $(n, T') \models \diamond\psi$.

Due to the well known equivalence (Theorem 1, for the case of least fixed points $\mu x.\psi$, we prove for the corresponding finite expansion $\bigvee_{i=0}^{k} \mu^i x.\psi$, where k is the number of nodes of the Kripke structure satisfying the input formula. By assumption we know there is some i, such that $n \vdash \mu^i x.\psi$. A plan $(n, T_1, \dots, T_m) \models \mu^i x.\psi$ is obtained by applying a second structural induction on $\mu^i x.\psi$. This proof is the same as the cases already proven above. Now, by Definition 5, we obtain $(n, T_1, \dots, T_m) \models \bigvee_{i=0}^{k} \mu^i x.\psi$.

Similarly as it is proven the least fixed point case, it is for the greatest fixed point. □

4 Conclusions

Branching path planning for mobile agents are studied in the setting of the modal μ-calculus. In contrast with linear plans, where tasks are performed in a linear sequential order, in branching plans, several linear plans can be performed concurrently and asynchronously by a team of agents. Since plans are tree shaped, it is then guaranteed agents never collide. Since the μ-calculus is one of the most expressive, yet computable, formalisms, complex specifications for agents can be denoted by logic formulas. In particular, specifications involving finite and infinite recursion are nicely capture by the least and greatest fixed-points of the μ-calculus. In contrast with traditional plan generation in the context of temporal logics, where plans are obtained from counterexample generation in model checking tools, in the current work, we develop a model checking algorithm based on a sequent derivation system. This system produces proof trees whenever a formula is satisfied by the model. Plans are then directly extracted from the proof trees with the use of logical flow graphs. We provide correctness proofs for both, model checking and plan generation.

The inference system proposed in the current work is a general approach for non-linear plan generation. It can then be applied in motion planning for mobile robots by means of well-known discrete abstraction of the continuous environments typically occurring in robotics applications. We are also interested in the study of the application of our proposal in the setting of context-aware navigation systems for mobile devices.

Other further immediate research perspectives concerns the study and development of implementation techniques for the sequent system provided in this paper. In particular, we are interested in the representation of models based in binary decision diagrams [19]. We also expect to extend the approach proposed in the current paper, for the case of more expressive logics involving arithmetical constraints [4, 3, 2].

Acknowledgment

This work was partially developed under the support of the National Council of Science and Technology (CONACYT) in the scope of the Cátedras CONACYT project "Infraestructura para Agilizar el Desarrollo de Sistemas Centrados en el Usuario" (Ref 3053)

References

1. **Bárcenas, E., Benitez, A., de la Calleja, J., Medina, M. A., & Ríos-Martínez, J. (2014).** Reasoning about the past on temporal specifications for motion planning. *CONIELECOMP*, IEEE, pp. 206–211.
2. **Bárcenas, E., Genevès, P., Layaïda, N., & Schmitt, A. (2011).** Query reasoning on trees with types, interleaving, and counting. **Walsh, T.**, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI/AAAI, pp. 718–723.
3. **Bárcenas, E. & Lavalle, J. (2013).** Expressive reasoning on tree structures: Recursion, inverse programs, Presburger constraints and nominals. **Espinoza, F. C., Gelbukh, A. F., & González-Mendoza, M.**, editors, *Advances in Artificial Intelligence and Its Applications - 12th Mexican International Conference on Artificial Intelligence, MICAI 2013, Proceedings, Part I*, volume 8265 of *Lecture Notes in Computer Science*, Springer, pp. 80–91.
4. **Bárcenas, E. & Lavalle, J. (2014).** Global numerical constraints on trees. *Logical Methods in Computer Science*, Vol. 10, No. 2.
5. **Blackburn, P., Benthem, J. F. A. K. v., & Wolter, F. (2006).** *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA.
6. **Bobadilla, L., Sanchez, O., Czarnowski, J., Gossman, K., & LaValle, S. M. (2011).** Controlling wild bodies using linear temporal logic. **Durrant-Whyte, H. F., Roy, N., & Abbeel, P.**, editors, *Robotics: Science and Systems VII*.

7. **Bonatti, P. A., Lutz, C., Murano, A., & Vardi, M. Y. (2008).** The complexity of enriched mu-calculi. *Logical Methods in Computer Science*, Vol. 4, No. 3.
8. **Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., & Hwang, L. J. (1990).** Symbolic model checking: 10ˆ20 states and beyond. *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, IEEE Computer Society, pp. 428–439.
9. **Buss, S. R. (1991).** The undecidability of k-provability. *Ann. Pure Appl. Logic*, Vol. 53, No. 1, pp. 75–102.
10. **Clarke, E. M. & Veith, H. (2003).** Counterexamples revisited: Principles, algorithms, applications. **Dershowitz, N.**, editor, *Verification: Theory and Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *Lecture Notes in Computer Science*, Springer, pp. 208–224.
11. **Conner, D. C., Rizzi, A. A., & Choset, H. (2003).** Composition of local potential functions for global robot control and navigation. *IROS*, IEEE, pp. 3546–3551.
12. **Dix, J., Posegga, J., & Schmitt, P. H. (1990).** Modal logics for AI planning. *Proceedings of the First International Conference on Expert Planning Systems, June 27-29, 1990*, AAAI, pp. 157–162.
13. **Fainekos, G. E., Girard, A., Kress-Gazit, H., & Pappas, G. J. (2009).** Temporal logic motion planning for dynamic robots. *Automatica*, Vol. 45, No. 2, pp. 343–352.
14. **Guo, M., Johansson, K. H., & Dimarogonas, D. V. (2013).** Revising motion planning under linear temporal logic specifications in partially known workspaces. *ICRA*, IEEE, pp. 5025–5032.
15. **Holzmann, G. J. (1997).** The model checker SPIN. *IEEE Trans. Software Eng.*, Vol. 23, No. 5, pp. 279–295.
16. **Kloetzer, M. & Belta, C. (2010).** Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, Vol. 26, No. 1, pp. 48–61.
17. **Lenzi, G. (2005).** The modal μ-calculus: a survey. *TASK Quaterly*, Vol. 9, No. 3, pp. 293–316.
18. **McCarthy, J. & Hayes, P. J. (1969).** Some philosophical problems from the standpoint of artificial intelligence. In **Meltzer, B. & Michie, D.**, editors, *Machine Intelligence 4*. Edinburgh University Press, pp. 463–502.
19. **Pan, G., Sattler, U., & Vardi, M. Y. (2006).** Bdd-based decision procedures for the modal logic K. *Journal of Applied Non-Classical Logics*, Vol. 16, No. 1-2, pp. 169–208.
20. **Rabin, M. O. (1968).** Decidability of second-order theories and automata on infinite trees. *Bull. Amer. Math. Soc.*, Vol. 74, No. 5, pp. 1025–1029.
21. **Seidel, R. (2010).** Reprint of: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom.*, Vol. 43, No. 6-7, pp. 556–564.
22. **Tarski, A. (1955).** A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, Vol. 5, No. 2, pp. 285–309.
23. **Ulusoy, A., Smith, S. L., Ding, X. C., Belta, C., & Rus, D. (2013).** Optimality and robustness in multi-robot path planning with temporal logic constraints. *I. J. Robotic Res.*, Vol. 32, No. 8, pp. 889–911.
24. **Wang, C., Hwang, R., & Ting, C. (2011).** Ubipapago: Context-aware path planning. *Expert Syst. Appl.*, Vol. 38, No. 4, pp. 4150–4161.
25. **Wolff, E. M., Topcu, U., & Murray, R. M. (2013).** Automaton-guided controller synthesis for nonlinear systems with temporal logic. *IROS*, IEEE, pp. 4332–4339.

Article received on 14/05/2015; accepted on 22/06/2016.
Corresponding author is Everardo Bárcenas.