Extensible Proof Systems for Infinite-State Systems

RANCE CLEAVELAND, Department of Computer Science, University of Maryland, USA
JEROEN J. A. KEIREN, Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

This article revisits soundness and completeness of proof systems for proving that sets of states in infinite-state labeled transition systems satisfy formulas in the modal mu-calculus in order to develop proof techniques that permit the seamless inclusion of new features in this logic. Our approach relies on novel results in lattice theory, which give constructive characterizations of both greatest and least fixpoints of monotonic functions over complete lattices. We show how these results may be used to reason about the sound and complete tableau method for this problem due to Bradfield and Stirling. We also show how the flexibility of our lattice-theoretic basis simplifies reasoning about tableau-based proof strategies for alternative classes of systems. In particular, we extend the modal mu-calculus with timed modalities, and prove that the resulting tableau method is sound and complete for timed transition systems.

CCS Concepts: • **Theory of computation** → **Timed and hybrid models**; **Modal and temporal logics**; **Verification by model checking**; *Proof theory*; *Logic;*

Additional Key Words and Phrases: Mu-calculus, model checking, infinite-state systems, timed systems, tableaux

ACM Reference format:
Rance Cleaveland and Jeroen J. A. Keiren. 2023. Extensible Proof Systems for Infinite-State Systems. *ACM Trans. Comput. Logic* 25, 1, Article 2 (November 2023), 60 pages.
https://doi.org/10.1145/3622786

1 INTRODUCTION

Proof systems provide a means for proving sequents in formal logics, and are intended to reduce reasoning about objects in a given theory to mechanically checkable arguments consisting of applications of proof rules to the sequents in question. When a proof system is sound, every provable sequent is indeed semantically valid; when it is complete, every semantically valid sequent can be proved within the proof system. Because proof rules in these systems manipulate syntax, the construction of proofs within them can be automated; proof assistants such as Coq [7] and Nuprl [21] are built around this observation. Within the model-checking community, the fully automatic construction of proofs based on sound and complete proof systems for decidable theories provides

Research supported by US National Science Foundation grant CNS-1446365 and US Office of Naval Research grant N00014-17-1-2622.
Authors' addresses: R. Cleaveland, Department of Computer Science, University of Maryland, College Park, Maryland 20742; e-mail: rance@cs.umd.edu; J. J. A. Keiren, Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, Eindhoven 5600 MB, The Netherlands; e-mail: j.j.a.keiren@tue.nl.

1529-3785/2023/11-ART2
https://doi.org/10.1145/3622786

a basis for establishing the correctness, or incorrectness, of systems *vis à vis* properties they are expected to satisfy. In addition, proof systems can be used as a basis for different approaches to model checking. In classical *global* model-checking techniques [17], one uses the proof rules to prove properties of states with respect to larger and larger subformulas of the given formula, until one shows that the start state(s) of the system either do, or do not, satisfy the original formula. In *local* [58], or *on-the-fly* [8] methods, in contrast, one uses the proof rules to conduct goal-directed reasoning from the start states and original formula, applying proof rules "in reverse" to generate subgoals that require proving in order for the original sequents to be true. The virtue of on-the-fly model checking is that proofs can often be completed, or be shown not to exist, without having to examine all states and all subformulas.

Driven by applications in model checking, proof systems have been developed for establishing that finite-state systems satisfy formulas captured in a very expressive temporal logic, the *(propositional) modal mu-calculus* [44]. These in turn have been used as a basis for efficient model-checking procedures for fragments of this logic [20]. For interesting fragments of the mu-calculus, global and on-the-fly techniques exhibit the same worst-case complexity [20, 52], so the early-termination feature of on-the-fly approaches does not incur additional overhead in the worst case.

Researchers have also developed sound and complete proof systems for infinite-state systems and the modal mu-calculus [11, 13]. In this case, the sequents, instead of involving single states and formulas in the logic as in the finite-state case, refer to potentially infinite sets of states. In general infinite-state model checking in the modal mu-calculus is undecidable; nevertheless, specialized proof systems for modifications of the so-called *alternation-free* fragment of the mu-calculus [25, 67] have been shown to lead to efficient on-the-fly model checkers for *timed automata* [5], a class of infinite-state systems whose model-checking problem is decidable.

Ideally, one should be able to prove soundness and completeness of these timed-automata proof systems using the general results for infinite-state systems and the modal mu-calculus. One should also be able to develop checkers beyond the alternation-free fragment so that more types of properties can be processed. However, there are several obstacles to this desirable state of affairs.

(1) The intricacy of the proof system in [11] means that modifications to it in essence require re-proofs of soundness and completeness from first principles.
(2) The proof systems for on-the-fly model checking of timed automata require several modifications to the modal mu-calculus, such as including modalities for reasoning about time and computable proof-termination criteria to enable detecting when a proof attempt is complete.
(3) For efficiency reasons, construction of proofs in the on-the-fly model checkers must also use different proof-construction strategies, which is an obstacle to applying reasoning from the infinite-state mu-calculus proof system to establishing the correctness of these procedures.

These issues have limited the practical application of the proof system in [11], although its theoretical contribution is, rightfully, very highly regarded.

In this article, the goal is to revisit the proof system for general infinite-state systems and the modal mu-calculus with a view toward developing new, extensible proofs of soundness and completeness that can be adapted to the inclusion of new features, such as real time, in the logic. Concretely, our contributions are the following.

We first introduce a new notion, called a *support structure*, into the theory of monotonic functions over the complete lattices corresponding to the powerset of a given set. These support structures permit the constructive characterizations, in a precise sense, of both the least and greatest fixpoints of such monotonic functions. In particular, to prove that an element is in the greatest fixpoint of a function f, it suffices to 'construct' a set containing that element, and prove that set is a post-fixpoint of f. Similarly, our support structures allow us to show an element is part of the least fixpoint if we can construct a set containing that element, along with a well-founded ordering on that set, satisfying a condition analogous to the set being a post-fixpoint. To the best of our knowledge these results, especially in the context of least fixpoints, are new. We then use these characterizations as a purely semantic characterization of the semantics of fixpoint formulas in the modal mu-calculus. This allows us to avoid the introduction of infinitary syntactic constructs such as ordinal unfoldings that are the basis for previous formulations of the modal mu-calculus [11, 13, 59]. Explicit reasoning involving the ordinals is restricted completely to our general results involving support structures.

We next consider a slightly modified version of the proof system of [11], and show that soundness and completeness of the proof system follow from our lattice-theoretical results. In particular, a syntactic ordering derived from [11, 13], the *extended dependency ordering*, is shown to induce support structures on sets of states associated with fixpoint formulas in successful proofs, and from this observation soundness of the proof system follows straightforwardly. In a similar way, given a semantic support structure induced by our lattice-theoretical results, we show how to construct a tableau whose extended-dependency ordering respects the given support structure. This establishes completeness. To facilitate the completeness proof, we also establish a novel notion related to well-founded induction in the context of mutually recursive fixpoints.

Finally, we show that these results permit extensions to the mu-calculus to be easily incorporated in a soundness- and completeness-preserving manner. They also simplify reasoning about new proof-termination criteria. To show this, we first show how a modification of the success criterion of the proof system, which changes how least-fixpoint formulas are unfolded, is trivially sound based on our earlier results. The resulting proof system is not complete in general, but is useful for automation purposes. Second, we consider a proof system for timed transition systems (TTSs), and a mu-calculus with two additional, timed modalities [25]. The proofs of soundness and completeness are, indeed, straightforward extensions of our earlier results. To the best of our knowledge, ours is also the first sound and complete proof system for checking whether sets of states satisfy formulas in a timed modal mu-calculus.

The rest of the article develops along the following lines. In the next section, we discuss the related work. In Section 3, we review mathematical preliminaries used in the rest of the article. The expert reader can safely skip this section on first reading, and use it for reference of the notation and basic definitions used in this article. We then state and prove our lattice-theoretical results in Section 4, while Section 5 introduces the syntax and semantics of the modal mu-calculus and establishes some properties that will prove useful later in the article. We present the proof system of [11] in Section 6. For readers intimately familiar with the mu-calculus and Bradfield and Stirling's proof system it is probably sufficient to skip or skim Sections 5 and 6. In Section 7, we show that the soundness of the proof system follows from the lattice-theoretical results, and we consider completeness in Section 8. In Sections 9 and 10, we illustrate how our new approach accommodates changes to the proof system that are needed for efficient on-the-fly model checking for timed automata and other decidable formalisms. Conclusions and future work are discussed in Section 11. For space reasons, proofs of some results are included in the appendix; others are sketched or elided. In the latter case, the interested reader may find full proofs of all lemmas and theorems in [39].

2 RELATED WORK

Tableaux for proving that a single state in different classes of labeled transition systems (LTSs) satisfies a formula in the mu-calculus have been widely studied in the literature. Larsen [47] describes a proof system for Hennessy-Milner logic [32] with recursion in the setting of so-called *image-finite* LTSs. The logic is similar to the mu-calculus, but does not allow alternation between fixpoints. Stirling and Walker [58] use tableaux to define a method for locally determining whether a state in a *finite-state* LTS satisfies a given mu-calculus formula. Inspired by Larsen's work, it uses fixpoint induction to accommodate reasoning about the fixpoints in the mu-calculus. Stirling and Walker furthermore introduce *definition lists* to facilitate reasoning about alternating fixpoints. In his tableaux method implemented in the Concurrency Workbench [19], Cleaveland [18] uses an appropriately updated set of hypotheses, instead of definition lists, to deal with fixpoints in the setting of finite-state LTSs. Winskel [65] presents a set of inference rules that, when recast as tableaux, are similar to Cleaveland's, but that syntactically modify the formulas to keep track of the hypotheses. Mader [49] describes an optimization of the tableaux method that reuses sub-tableaux. Sokolsky and Smolka [53] give incremental algorithms for model-checking in the alternation-free mu-calculus. A method to extract counterexamples and witnesses from tableaux was described by Kick [40]. Tan and Cleaveland [61] and Cranen et al. [22] describe generic ways to store evidence in the context of mu-calculus model checkers.

The approaches for finite-state LTSs are generalized to proving that potentially infinite sets of states in infinite-state systems satisfy mu-calculus formulas by Bradfield and Stirling [10, 11, 13]. Similar to the finite-state tableaux of [58], the tableaux in these works use definition lists to terminate the unfolding of fixpoint formulas as the proofs of sequents are constructed. In particular, if a sequent is encountered whose set of states is a subset of a sequent labeled with the same definitional constant appearing earlier in the proof, no unfolding performed on this sequent. This immediately leads to success in case of greatest fixpoints. Success of least fixpoints is more involved, and requires that a specific ordering on states in the companion node is well-founded. This ordering is inferred syntactically from local state/subformula dependencies using so-called *trails* [10, 11] and is also known as the *extended dependency ordering* [13]. The local dependencies and the extended dependency ordering are, in turn, closely related to Streett and Emerson's derivation and regeneration relations, respectively [59]. A prototype implementation of the proof system for infinite-state LTSs is described in [14]. Anderson [6] describes a set of inference rules in the spirit of [65] that keep track of sets of states inside the formula, instead of using definition lists. These inference rules explicitly provide a well-founded order on states satisfying least fixpoints, and are readily translated into derivation rules in a tableau. Another way to explicitly give a well-founded order on states is to associate a *progress measure*, as introduced by Klarlund, to every state [41]. Along with a progress relation that requires the measure to decrease, a progress measure allows for locally checking that progress is made towards some objective such as fair termination [43] or a so-called *Liminf condition* [42].

Tableaux represent just one of the methods used to address the model-checking problem for the mu-calculus. The problem is also equivalent to checking emptiness of nondeterministic parity automata on infinite trees [24], and to solving parity games [23] or Boolean equation systems [50]. For thorough overviews of these results and other related work, as well as an in-depth discussion of model checking the modal mu-calculus, we refer the reader to overviews such as [12, 28, 55].

Tableaux for satisfiability and validity of the propositional modal mu-calculus have also been widely studied. The first such proof system is described by Kozen [44]; in that article he proves soundness and completeness for the so-called aconjunctive fragment of the mu-calculus. Completeness of an alternative proof system for the guarded fragment of the mu-calculus is proved by Walukiewicz [63]. Later, the same author also establishes completeness of Kozen's original system [64] for the same fragment. Jungteerapanich [38] presents cut-free proof systems using names for fixpoint formulas that work for the full mu-calculus. Stirling [56, 57] describes proof systems for validity similar to those by Jungteerapanich. The proof systems mentioned so far assume guarded mu-calculus formulas (i.e., every recursion appears in the scope of a modality). In terms of expressiveness this is not a restriction, since any mu-calculus formula can be translated into guarded form, although at the cost of an exponential blow-up [15]. Friedmann and Lange [27] present a tableau based on a new fixpoint unfolding rule that does not require this guardedness assumption. Jäger et al. describe two infinitary proof systems, $K_\omega^+(\mu)$ and $K_\omega(\mu)$, for the modal mu-calculus [35] whose completeness results are established without automata-theoretic arguments. They use these proof systems to derive a cut-free and complete finite system for the same logic. Afshari and Leigh [4] present a finitary proof system that is equivalent to infinitary proof system $K_\omega(\mu)$ from [35]; Studer [60] provides a transformation that can be used to embed the latter into Stirling's proof system [57] as shown in [4]. Marti and Venema [51] show that the approach from [38, 57] can be simplified when one only considers the alternation-free fragment of the mu-calculus. In particular, definition lists are replaced with an annotation of formulas with a single bit of information, the *focus*.

The results discussed so far pertain to the propositional modal mu-calculus. Several extensions and variations of this logic have been described in the literature. For instance, formulas can be presented equationally [20]. There are also several extensions of the mu-calculus to timed systems. Aceto and Laroussinie present the logic $L_{\mu,\nu}^+$ that extends the mu-calculus with freeze quantification and timed modalities $\exists$ and $\forall$ (analogous to the modalities $\langle - \rangle\phi$ and $[-]\phi$, respectively) [3]. Sokolsky and Smolka give an incremental algorithm for checking whether timed automata satisfy formulas in what is essentially the alternation-free fragment of $L_{\mu,\nu}^+$ [54]. A restriction, L_ν, of $L_{\mu,\nu}^+$ in which only greatest fixpoints are allowed, was studied by Laroussinie et al. [46]. The logic L_c extends L_ν with more expressive timed modalities. Model checking and expressiveness of this logic were examined in [9]. Larsen et al. studied the logic L_s, which restricts the use of modalities $\langle - \rangle\phi$, $\exists\phi$ and disjunctions in L_ν [48]. *SBBL* extends L_s by allowing $\langle a \rangle$*true* [2], and $L_{\forall S}$ extends L_s with an operator $\forall_S$, with S a set of actions [1]. Henzinger et al. present the logic T_μ that extends the mu-calculus with freeze quantification and a "next" operator $\triangleright$ [33]. Fontana and Cleaveland introduce relativized timed modal operators, akin to the timed until and release operators, into the mu-calculus [25].

3 MATHEMATICAL PRELIMINARIES

This section defines basic terminology used in the sequel for finite sequences, (partial) functions, binary relations, lattices and fixpoints, and finite trees.

3.1 Sequences

Sequences are ordered collections of elements $x_1 \cdots x_n$, where each x_i is taken from a given set X.

Notation 3.1 (Sequences). Let X be a set. We write X^* for the set of finite, possibly empty, sequences of elements from X. The *empty sequence* in X^* is denoted ε. We take $X \subseteq X^*$, with each $x \in X$ being a single-element sequence in X^*. Suppose $\vec{w} = x_1 \cdots x_n \in X^*$, where each $x_i \in X$. Then $|\vec{w}| = n$ denotes the *length* of $\vec{w}$. Note that $|\varepsilon| = 0$, and $|\vec{w}| = 1$ if $\vec{w} = x$ for some $x \in X$. If $\vec{w}_1, \vec{w}_2 \in X^*$ then $\vec{w}_1 \cdot \vec{w}_2 \in X^*$ is the *concatenation* of $\vec{w}_1$ and $\vec{w}_2$. We often omit $\cdot$ and write e.g., $\vec{w}_1\vec{w}_2$ for $\vec{w}_1 \cdot \vec{w}_2$. For $\vec{w}_1, \vec{w}_2 \in X^*$ we write $\vec{w}_1 \leq \vec{w}_2$ if $\vec{w}_1$ is a (not necessarily strict) *prefix* of $\vec{w}_2$, and $\vec{w}_1 \not\leq \vec{w}_2$ if $\vec{w}_1$ is not a prefix of $\vec{w}_2$. For $\vec{w} = x_1 \cdots x_n \in X^*$ the *set associated with* $\vec{w}$, $set(\vec{w}) \subseteq X$, is defined to be $\{x_1, \ldots, x_n\}$. Note that $set(\varepsilon) = \emptyset$, and $|set(\vec{w})| \leq |\vec{w}|$. Sequence $\vec{w} \in X^*$ is *duplicate-free* iff $|\vec{w}| = |set(\vec{w})|$, and it is a *permutation*, or *ordering*, of X iff $\vec{w}$ is duplicate-free and $set(\vec{w}) = X$.

Note that only finite sets can have permutations/orderings in this definition.

3.2 Partial Functions

In this article, we make significant use of partial as well as total functions. This section introduces our notational conventions for such functions.

Notation 3.2 (Partial Functions). Let X and Y be sets.

- Relation $f \subseteq X \times Y$ is *functional* iff for all $x \in X$ and $y_1, y_2 \in Y$, if $(x, y_1) \in f$ and $(x, y_2) \in f$ then $y_1 = y_2$. When f is functional we call f a *partial function* from X to Y. We use $X \rightarrow_\perp Y$ to denote the set of all partial functions from X to Y.
- Suppose $f \in X \rightarrow_\perp Y$ and $x \in X$. If there is $y \in Y$ such that $(x, y) \in f$ then we write $f(x)$ as usual to denote this y and say f is *defined* for x in this case. We will also write $f(x) \in Y$ to denote that f is defined for x. If there exists no $y \in Y$ such that $(x, y) \in f$ then we say that f is *undefined* for x and write $f(x)\perp$.
- If $f \in X \rightarrow_\perp Y$ then we call $\text{dom}(f) = \{x \in X \mid f(x) \in Y\}$ the *domain of definition* of f.
- $f \in X \rightarrow_\perp Y$ is *total* iff $\text{dom}(f) = X$. We write $X \rightarrow Y$ as usual for the set of total functions from X to Y. Note that $X \rightarrow Y \subseteq X \rightarrow_\perp Y$.
- If $f, g \in X \rightarrow_\perp Y$ then $f = g$ iff $\text{dom}(f) = \text{dom}(g)$ and for all $x \in \text{dom}(f)$, $f(x) = g(x)$.

Partial functions are equal exactly when they are defined on the same elements and return the same values when they are defined. We also use the following standard operations on partial functions.

Notation 3.3 (Function Operations). Let X and Y be sets.

- Let $f \in X \rightarrow_\perp X$ and $i \in \mathbb{N}$. Then $f^i \in X \rightarrow_\perp X$ is defined as follows: $f^i(x) = x$ if $i = 0$ and $f(f^{i-1}(x))$ otherwise. Note that f^0 is total and that $f^i(x)\perp$ iff $f(f^j(x))\perp$ some $j < i$. Also note that if f is total then so is f^i for all $i \geq 0$.
- Suppose $f \in X \rightarrow_\perp Y$, $x \in X$ and $y \in Y$. Then $f[x := y] \in X \rightarrow Y$ is defined as follows: $f[x := y](x') = y$ if $x' = x$, and $f[x := y](x') = f(x')$ if $x' \neq x$ and $f(x') \in Y$. Note $f[x := y]$ is defined for x even if $f(x)\perp$, and that if f is total then so is $f[x := y]$. This notion can generalized to $f[\overrightarrow{x} := \overrightarrow{y}]$, where $\overrightarrow{x} \in X^*$ is duplicate-free and $\overrightarrow{y} \in Y^*$ is such that $|\overrightarrow{x}| = |\overrightarrow{y}|$, in the obvious fashion.
- Let $f \in X \rightarrow_\perp Y$ and $\overrightarrow{w} = x_1 \cdots x_n \in X^*$. Then $f \in X^* \rightarrow_\perp Y^*$ is defined to by $f(\overrightarrow{w}) = f(x_1) \cdots f(x_n)$. Note that if $f(x)\perp$ for any $x \in \mathit{set}(\overrightarrow{w})$ then $f(\overrightarrow{w})\perp$.

3.3 Binary Relations

In this article, we make extensive use of the theory of *binary relations*, and we summarize some of the necessary concepts in this section.

If X is a set, then a *binary relation* over X is a subset $R \subseteq X \times X$. When R is a binary relation over X we usually write $x_1 \mathrel{R} x_2$ in lieu of $(x_1, x_2) \in R$ and $x_1 \not\mathrel{R} x_2$ instead of $(x_1, x_2) \notin R$. We now recall the following terminology.

Definition 3.4 (Preorders, Partial Orders and Equivalence Relations). Let $R \subseteq X \times X$ be a binary relation over X. R is *reflexive* iff $x \mathrel{R} x$ for all $x \in X$. R is *symmetric* iff whenever $x_1 \mathrel{R} x_2$ then $x_2 \mathrel{R} x_1$. R is *anti-symmetric* iff whenever $x_1 \mathrel{R} x_2$ and $x_2 \mathrel{R} x_1$ then $x_1 = x_2$. R is *transitive* iff whenever $x_1 \mathrel{R} x_2$ and $x_2 \mathrel{R} x_3$ then $x_1 \mathrel{R} x_3$. R is a *preorder* iff R is reflexive and transitive. A preorder that is anti-symmetric, is a *partial order*. A symmetric preorder is called an *equivalence relation*.

We also use the following standard, if less well-known, definitions.

Definition 3.5 (Irreflexive and Total Relations). Let $R \subseteq X \times X$. Relation R is *irreflexive* iff for every $x \in X$, $x \not\mathrel{R} x$. Relation R is a *(strict) total order* iff it is irreflexive and transitive and satisfies: for all $x_1 \neq x_2 \in X$, either $x_1 \mathrel{R} x_2$ or $x_2 \mathrel{R} x_1$.

A relation R over X is irreflexive iff no element in X is related to itself. It is total exactly when it is irreflexive and transitive and any distinct $x_1, x_2 \in X$ are *comparable*, one way or another, via R. This version of totality is often called *strict totality*, although we drop the qualifier "strict" in this article. The following relations are used later.

Definition 3.6 (Identity, Universal Relations). Let X be a set. The *identity relation over* X is defined as $Id_X = \{(x, x) \mid x \in X\}$. The *universal relation over* X is defined as $U_X = X \times X$.

We also use the following notions on binary relations.

Definition 3.7 (Relational Terminology). Let R, R' be binary relations over X. R' *extends* R iff $R \subseteq R'$. For $X' \subseteq X$, the *restriction* of R with respect to X' is the binary relation $R\lfloor X'$ over X' defined as $R\lfloor X' = R \cap U_{X'} = \{(x_1, x_2) \in X' \times X' \mid x_1 \; R \; x_2\}$. The *relational composition* of R and R' is the binary relation $R;R'$ over X defined as $R;R' = \{(x_1, x_3) \in X{\times}X \mid \exists x_2 \in X\colon x_1 \; R \; x_2 \wedge x_2 \; R' \; x_3\}$. The *inverse* of R is the binary relation R^{-1} over X defined by $R^{-1} = \{(x_2, x_1) \in X{\times}X \mid x_1 \; R \; x_2\}$. If $X' \subseteq X$ then the *image*, $R(X')$, of R with respect to X' is defined by $R(X') = \{x \in X \mid \exists x' \in X'\colon x' \; R \; x\}$. If $x \in X$ then we write $R(x)$ in lieu of $R(\{x\})$. Similarly, the *pre-image* of R with respect to X' is the image $R^{-1}(X')$ of R^{-1} with respect to X'. The *irreflexive core* of R is the binary relation R^- over X defined by $R^- = R \setminus Id_X$. The *reflexive closure* of R is the binary relation $R^=$ given by $R^= = R \cup Id_X$. The transitive closure, R^+, of R is the least transitive relation extending R. The reflexive and transitive closure, R^* of R, is defined by $R^* = (R^+)^=$.

Relation R^+ is guaranteed to exist for arbitrary set X and relation R over X, and from the definition it is immediate that R itself is transitive iff $R^+ = R$. In addition, for any relation $R \subseteq X \times X$, relation R^* is the unique smallest preorder that extends R. The reflexive and transitive closure of a relation induces an associated equivalence relation and partial order over the resulting equivalence classes, as the next definition indicates.

Definition 3.8 (Quotient of a Relation). Let $R \subseteq X \times X$ be a relation. Relation $\sim_R$ is defined as $x_1 \sim_R x_2$ iff $x_1 \; R^* \; x_2$ and $x_2 \; R^* \; x_1$. If $x \in X$, the equivalence class of x is defined as $[x]_R = \{x' \in X \mid x \sim_R x'\}$. We use $Q_R = \{\, [x]_R \mid x \in X\}$. The *ordering* $P(R) \subseteq Q_R \times Q_R$ *induced by* R *on* Q_R, is defined as $P(R) = \{([x]_R, [x']_R) \mid x \; R^* \; x'\}$; $(Q_R, P(R))$ is called the *quotient* of R.

It is easy to verify that $\sim_R$ is an equivalence relation for any R, that $[x]_R$ is the equivalence class of x with respect to $\sim_R$, and that $P(R)$ is a partial order over Q_R. Note that Q_R defines a partition of set X: $X = \bigcup_{Q \in Q_R} Q$, and either $Q = Q'$ or $Q \cap Q' = \emptyset$ for any $Q, Q' \in Q_R$.

In this article, we also make extensive use of *well-founded relations* and *well-orderings*. To define these, we first introduce the following, where we write $x \neq x' \in X'$ when $x, x' \in X'$ and $x \neq x'$.

Definition 3.9 (Extremal Elements). Let $R \subseteq X{\times}X$, and let $X' \subseteq X$. Element $x' \in X'$ is *R-minimal in* X' iff for all $x \neq x' \in X'$, $x \not R \; x'$. If it is the only R-minimal element in X', x' is *R-minimum in* X'. Likewise, $x' \in X'$ is *R-maximal in* X' iff for all $x \neq x' \in X'$, $x' \not R \; x$. If it is the only R-maximal element in X', x' is *R-maximum in* X'.

Element $x \in X$ is an *R-lower bound* of X' iff for all $x' \in X'$, $x \; R \; x'$. If x is R-maximum of the R-lower bounds of X' it is the *R-greatest lower bound* of X'. Similarly, $x \in X$ is an *R-upper bound* of X' iff for all $x' \in X'$, $x' \; R \; x$. If x is the R-minimum of the R-upper bounds of X' it is the *R-least upper bound* of X'.

In what follows we often omit R when it is clear from the context and instead write minimal rather than R-minimal, and so forth. Note that minimal/minimum/maximal/maximum elements for X' must themselves belong to X'; this is not the case for upper and lower bounds. We can now define well-founded relations and well-orderings.

Definition 3.10 (Well-founded Relations, Well-orderings). Let $R \subseteq X \times X$. R is *well-founded* iff every non-empty $X' \subseteq X$ has an R-minimal element. R is *well-ordering* iff it is total and well-founded.

We close the section by remarking on some noteworthy properties of well-founded relations and well-orderings.[1] The first result is a well-known alternative characterization of well-foundedness. If X is a set and $R \subseteq X \times X$, call $\ldots, x_2, x_1$ an *infinite descending chain in R* iff for all $i \geq 1$, $x_{i+1} \; R \; x_i$.

Lemma 3.11 (Descending Chains and Well-foundedness). *Let $R \subseteq X \times X$. Then R is well-founded iff R contains no infinite descending chains.*

The next lemma asserts that transitive closures preserve well-foundedness, and well-founded relations can be extended to well-orderings. Note that if $R = \emptyset$ the second statement reduces to the Well-Ordering Theorem [31], which states that every set can be well-ordered.

Lemma 3.12 (Transitive Closures and Total Extensions of Well-founded Relations). *Let $R \subseteq X \times X$ be well-founded. Then R^+ is well-founded, and there exists a well-ordering $R' \subseteq X \times X$ extending R.*

The next result is immediate from the definition of well-ordering.

Lemma 3.13 (Minimum Elements and Well-orderings). *Let $R \subseteq X \times X$ be a well-ordering. Then every non-empty $X' \subseteq X$ contains an R-minimum element.*

3.4 Complete Lattices, Monotonic Functions, and Fixpoints

This article relies heavily on the theory of fixpoints of monotonic functions over complete lattices, as developed by Tarski and Knaster [62]. We review the relevant parts of the theory here.

Definition 3.14 (Complete Lattice). A *complete lattice* is a tuple $(X, \sqsubseteq, \bigsqcup, \bigsqcap)$ where: X is a set (the *carrier set*); relation $\sqsubseteq$ is a partial order over X; function $\bigsqcup \in 2^X \to X$, the *join* operation, satisfies: for all $X' \subseteq X$, $\bigsqcup(X')$ is the least upper bound of X'; and function $\bigsqcap \in 2^X \to X$, the *meet* operation, satisfies: for all $X' \subseteq X$, $\bigsqcap(X')$ is the greatest lower bound of X'.

In what follows we write $\bigsqcup X'$ and $\bigsqcap X'$ instead of $\bigsqcup(X')$ and $\bigsqcap(X')$.

Definition 3.15 (Fixpoint). Let X be a set and $f \in X \to X$ be a function. Then $x \in X$ is a *fixpoint* of f iff $f(x) = x$.

As usual, $f \in X \to X$ is monotonic over complete lattice $(X, \sqsubseteq, \bigsqcup, \bigsqcap)$ iff whenever $x_1 \sqsubseteq x_2$, $f(x_1) \sqsubseteq f(x_2)$. The next result follows from the Tarski-Knaster Fixpoint Theorem [62].

Lemma 3.16 (Extremal Fixpoint Characterizations). *Let $(X, \sqsubseteq, \bigsqcup, \bigsqcap)$ be a complete lattice, and let $f \in X \to X$ be monotonic over this lattice. Then f has least and greatest fixpoints $\mu f, \nu f \in X$, respectively, characterized as follows:*

$$\mu f = \bigsqcap \{x \in X \mid f(x) \sqsubseteq x\},$$
$$\nu f = \bigsqcup \{x \in X \mid x \sqsubseteq f(x)\}.$$

Elements $x \in X$ such that $f(x) \sqsubseteq x$ are sometimes called *pre-fixpoints* of f, while those satisfying $x \sqsubseteq f(x)$ are referred to as *post-fixpoints* of f.

In this article, we focus on specialized complete lattices called *subset lattices.*

Definition 3.17 (Subset Lattice). The *subset lattice generated by set S* is the tuple $(2^S, \subseteq, \bigcup, \bigcap)$.

Note that for any S, the subset lattice generated by S is indeed a complete lattice.

[1]These results generally rely on the inclusion of additional axioms beyond the standard ones of **Zermelo–Fraenkel** (**ZF**) set theory. The Axiom of Choice [37] is one such axiom, and in the rest of the article we assume its inclusion in ZF.

3.5 Finite Trees

The proof objects in this article are finite trees whose nodes are labeled by sequents. In order to reason about mathematical constructions on these proof objects we must define such trees formally.

Definition 3.18 (Finite Non-empty Ordered Tree). A *finite non-empty ordered tree* is a tuple $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, where: $\mathbf{N}$ is a finite, non-empty set of *nodes*; $\mathbf{r} \in \mathbf{N}$ is the *root* node; $p \in \mathbf{N} \to_{\perp} \mathbf{N}$, the (partial) *parent* function, satisfies: $p(\mathbf{n})\perp$ iff $\mathbf{n} = \mathbf{r}$, and for all $\mathbf{n} \in \mathbf{N}$ there exists $i \geq 0$ such that $p^i(\mathbf{n}) = \mathbf{r}$; and $cs \in \mathbf{N} \to \mathbf{N}^*$, the *child-ordering* function, satisfies: for all $\mathbf{n} \in \mathbf{N}$, $cs(\mathbf{n}) \in \mathbf{N}^*$ is an ordering of $\{\mathbf{n}' \in \mathbf{N} \mid p(\mathbf{n}') = \mathbf{n}\}$.

In tree $\mathbf{T} = (\mathbf{n}, \mathbf{r}, p, cs)$ each non-root node $\mathbf{n} \neq \mathbf{r}$ has a parent node $p(\mathbf{n}) \in \mathbf{N}$. If $p(\mathbf{n}') = \mathbf{n}$ then we call $\mathbf{n}'$ a *child* of $\mathbf{n}$; we use $c(\mathbf{n}) = \{\mathbf{n}' \in \mathbf{N} \mid p(\mathbf{n}') = \mathbf{n}\}$ to denote all the children of $\mathbf{n}$. It can be seen that $cs(\mathbf{n})$ is an ordering on $c(\mathbf{n})$, with the elements of $c(\mathbf{n})$ listed in left-to-right order in $cs(\mathbf{n})$. If $c(\mathbf{n}) = \emptyset$ (or equivalently, $cs(\mathbf{n}) = \varepsilon$) then $\mathbf{n}$ is a *leaf*; otherwise, it is *internal.* We call node $\mathbf{n}$ an *ancestor* of node $\mathbf{n}'$, and $\mathbf{n}'$ a *descendant* of $\mathbf{n}$, iff there exists an $i \geq 0$ such that $p^i(\mathbf{n}') = \mathbf{n}$; we also say in this case that there is a *path from* $\mathbf{n}$ *to* $\mathbf{n}'$. We write $A(\mathbf{n})$ and $D(\mathbf{n})$ for the ancestors and descendants of $\mathbf{n}$, respectively, and note that $\mathbf{r} \in A(\mathbf{n})$, $\mathbf{n} \in D(\mathbf{r})$, $\mathbf{n} \in A(\mathbf{n})$ and $\mathbf{n} \in D(\mathbf{n})$ for all $\mathbf{n} \in \mathbf{N}$. We write $\mathbf{T_n}$ for the subtree of $\mathbf{T}$ rooted at node $\mathbf{n}$. The definition of subtree is standard. We use $A{\uparrow}(\mathbf{n}) = A(\mathbf{n}) \setminus \{\mathbf{n}\}$ and $D{\downarrow}(\mathbf{n}) = D(\mathbf{n}) \setminus \{\mathbf{n}\}$ for the *strict* ancestors and descendants of $\mathbf{n}$.

Finite ordered trees admit the following induction and co-induction principles.

Principle 3.19 (Tree Induction). Let $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ be a finite ordered tree, and let Q be a predicate over $\mathbf{N}$. To prove that $Q(\mathbf{n})$ holds for every $\mathbf{n} \in \mathbf{N}$, it suffices to prove $Q(\mathbf{n})$ under the assumption that $Q(\mathbf{n}')$ holds for every $\mathbf{n}' \in D{\downarrow}(\mathbf{n})$. The assumption is referred to as the *induction hypothesis.*

Principle 3.20 (Tree Co-induction[2]). Let $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ be a finite ordered tree, and let Q be a predicate over $\mathbf{N}$. To prove that $Q(\mathbf{n})$ holds for every $\mathbf{n} \in \mathbf{N}$, it suffices to prove that $Q(\mathbf{n})$ holds under the assumption that $Q(\mathbf{n}')$ holds for every $\mathbf{n}' \in A{\uparrow}(\mathbf{n})$. The assumption is referred to as the *co-induction hypothesis.*

Tree induction is an instance of standard strong induction on the height of nodes, while tree co-induction corresponds to standard strong induction on the depth of nodes, where node height and depth are defined in the usual manner, with the height of leaves and the depth of the root both taken to be 0. Note that the co-induction principle also applies to discrete rooted infinite trees [36, 45] as well as finite ones, although we do not need this fact for this article. Since the root node $\mathbf{r}$ is the only node with no strict ancestors, in the co-inductive arguments given later we will often single out a special *root case* for dealing with $\mathbf{r}$, with reasoning about other nodes covered in a so-called *co-induction step.*

4 SUPPORT STRUCTURES AND FIXPOINTS

A key contribution of this article is a novel characterization of least fixpoints for monotonic functions over subset lattices. In contrast with the least-fixpoint result in Lemma 3.16 this characterization may be seen as constructive in a precise sense, and relies on the notion of *support structure.*

[2]Accounts of co-induction tend to focus on its use in reasoning about co-algebras. The treatment of finite trees in this article is not explicitly co-algebraic, but the principle of co-induction as articulated in e.g., [34] is easily seen to correspond to what is given here.

Definition 4.1 (Support Structure). Let S be a set, let $(2^S, \subseteq, \bigcup, \bigcap)$ be the subset lattice generated by S, and let $f \in 2^S \rightarrow 2^S$ be a monotonic function over this lattice. Then $(X, \prec)$ is a *support structure* for f iff the following hold.

(1) $X \subseteq S$ and $\prec \subseteq X \times X$ is a binary relation on X.
(2) For all $x \in X$, $x \in f(\prec^{-1}(x))$.[3]

Intuitively, a support structure $(X, \prec)$ for monotonic function f contains sufficient information, in the form of $\prec^{-1}(x)$ for each $x \in X$, to determine that each $x \in X$ is also in the set $f(X)$.

We call a support structure $(X, \prec)$ for monotonic f *well-founded* if $\prec$ is well-founded and say X is *(well-)supported* for f in this case.

Example 4.2. Consider the subset lattice $(2^{\mathbb{N}}, \subseteq, \bigcup, \bigcap)$ over the set $\mathbb{N}$ of natural numbers, and let $<$ be the standard ordering on elements of $\mathbb{N}$. Also let $f \in 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ be defined as

$$f(S) = \{0\} \cup \{n + 1 \mid n \in S\}.$$

Note that f is monotonic with respect to the subset ordering $\subseteq$: if $S_1 \subseteq S_2 \subseteq \mathbb{N}$ then $f(S_1) \subseteq f(S_2)$. It can be seen that both $(\{0\}, R_0)$, where $R_0 = (<)\lfloor\{0\}$ is the relation $<$ restricted to the set $\{0\}$, and $(\mathbb{N}, <)$ are well-founded support structures for f. Clearly both R_0 and $<$ are well-founded. To see that the first is a support ordering, note that $R_0^{-1}(0) = \emptyset$ and $f(\emptyset) = \{0\}$, so $0 \in f(R_0^{-1}(0))$. For the latter, it can be seen that for any $n \in \mathbb{N}$, $<^{-1}(n) = \{0, 1, \ldots, n-1\}$, and that $n \in f(<^{-1}(n)) = \{0, 1, \ldots, n\}$. On the other hand, $(\{1, 2\}, R_{12})$, where $R_{12} = (<)\lfloor\{1, 2\}$ is the restriction of relation $<$ to the set $\{1, 2\}$, is not a well-founded support structure for f even though R_{12} is well-founded. To see why, note that $R_{12}^{-1}(1) = \emptyset$, and $1 \notin f(\emptyset) = \{0\}$.

The support-ordering characterization of the least fixpoint μf of monotonic function f over a given subset lattice relies on the following lemma, which asserts that the union of a collection of well-supported subsets of S is also well-supported.

Lemma 4.3 (Unions of Well-supported Sets). *Let S be a set, and let $f \in 2^S \rightarrow 2^S$ be monotonic over subset lattice $(2^S, \subseteq, \bigcup, \bigcap)$. Also let $\mathcal{W} \subseteq 2^S$ be a set of well-supported sets for f. Then $\bigcup \mathcal{W}$ is well-supported for f.*

Proof sketch. The idea of the proof is as follows. We first order $\mathcal{W}$ arbitrarily, but in a well-founded manner. A well-founded relation for $\bigcup \mathcal{W}$ is then constructed by taking the elements that are less that $x \in \bigcup \mathcal{W}$ from the first set in $\mathcal{W}$ containing x according to the ordering. The resulting relation is then shown to induce a well-founded support structure on $\bigcup \mathcal{W}$. The detailed proof is included in the appendix. □

We now have the following:

Theorem 4.4. *Let S be a set, and let $f \in 2^S \rightarrow 2^S$ be a monotonic function over the subset lattice $(2^S, \subseteq, \bigcup, \bigcap)$.*

(1) For all $X \subseteq S$, if X is well-supported for f then $X \subseteq \mu f$.
(2) Let $\mathcal{X} = \{X \subseteq S \mid X$ is well-supported for $f\}$. Then $f(\bigcup \mathcal{X}) = \bigcup \mathcal{X}$.

Proof. To prove statement (1), suppose $X \subseteq S$ is well-supported for f, and let $\prec \subseteq X \times X$ be a well-founded relation such that $(X, \prec)$ is a support structure for f. Also recall that $\mu f = \bigcap\{Y \subseteq S \mid f(Y) \subseteq Y\}$. To show that $X \subseteq \mu f$ it suffices to show that $X \subseteq Y$ for all Y such that $f(Y) \subseteq Y$. So fix such a Y; we prove that for all $x \in X$, $x \in Y$ using well-founded induction on $\prec$. So fix $x \in X$.

[3]Recall that $\prec^{-1}(x) = \{x' \in X \mid x' \prec x\}$.

The induction hypothesis states that for all $x' \prec x$, $x' \in Y$. By definition of support structure we know that $x \in f(\prec^{-1}(x))$; the induction hypothesis also guarantees that $\prec^{-1}(x) \subseteq Y$. Since f is monotonic, $f(\prec^{-1}(x)) \subseteq f(Y)$, and thus we have $x \in f(\prec^{-1}(x)) \subseteq f(Y) \subseteq Y$. Hence $x \in Y$.

As for statement (2), Lemma 4.3 guarantees that $\bigcup \mathcal{X}$ is well-supported; let $\prec$ be a well-founded relation over $\bigcup \mathcal{X}$ such that $(\bigcup \mathcal{X}, \prec)$ is a support structure for f. It suffices to show that $f(\bigcup \mathcal{X}) \subseteq \bigcup \mathcal{X}$ and $\bigcup \mathcal{X} \subseteq f(\bigcup \mathcal{X})$. For the former, by way of contradiction assume x is such that $x \in f(\bigcup \mathcal{X})$ and $x \notin \bigcup \mathcal{X}$. Now consider the relation $\prec'$ on $(\bigcup \mathcal{X}) \cup \{x\}$ given by: $\prec' = \prec \cup \{(x', x) \mid x' \in \bigcup \mathcal{X}\}$. It is easy to see that $\prec'$ is well-founded, and that $((\bigcup \mathcal{X}) \cup \{x\}, \prec')$ is a support structure for f. This implies $(\bigcup \mathcal{X}) \cup \{x\} \subseteq \bigcup \mathcal{X}$, which contradicts the assumption that $x \notin \bigcup \mathcal{X}$. To see that $\bigcup \mathcal{X} \subseteq f(\bigcup \mathcal{X})$, note that, since $(\bigcup \mathcal{X}, \prec)$ is a support structure, $x \in f(\prec^{-1}(x))$ and $\prec^{-1}(x) \subseteq \bigcup \mathcal{X}$ for all $x \in \bigcup \mathcal{X}$. Since f is monotonic, we have $x \in f(\prec^{-1}(x)) \subseteq f(\bigcup \mathcal{X})$ for all $x \in \bigcup \mathcal{X}$. □

The following corollary provides a *constructive* characterization of μf in the following sense: to establish that $x \in \mu f$ it suffices to construct a well-supported set X for f such that $x \in X$.

Corollary 4.5. *Let f be a monotonic function over the subset lattice generated by S. Then $\mu f = \bigcup\{X \in 2^S \mid X$ is well-supported for $f\}$.*

Support structures can also be used to characterize νf, the greatest fixpoint of monotonic function f. This characterization relies on the following observations.

Theorem 4.6. *Let S be a set, let $f \in 2^S \rightarrow 2^S$ be a monotonic function over the subset lattice $(2^S, \subseteq, \bigcup, \bigcap)$, and let $X \subseteq S$. Then X is supported for f iff $X \subseteq f(X)$.*

Proof. Let $X \subseteq S$ and f be given. For the only if direction, assume X is supported for f. This means there is a $\prec$ such that $(X, \prec)$ is a support structure for f. To show that $X \subseteq f(X)$, fix $x \in X$. Since $(X, \prec)$ is a support structure for f, $x \in f(\prec^{-1}(x))$. Since f is monotonic and $\prec^{-1}(x) \subseteq X$, we have $x \in f(X)$.

For the if direction, assume $X \subseteq f(X)$. We must come up with $\prec \subseteq X \times X$ such that $(X, \prec)$ is a support structure for f. Consider $\prec = U_X = X \times X$. Since for every $x \in X, x \in f(X)$ and $\prec^{-1}(x) = X$, we have that for every $x \in X, x \in f(\prec^{-1}(x))$, and $(X, \prec)$ is thus a support structure for f. □

This theorem establishes that any $X \subseteq S$ is a post-fixpoint for monotonic f (i.e., $X \subseteq f(X)$) iff X is supported for f. We also know from Lemma 3.16 that $\nu f = \bigcup\{X \subseteq S \mid X \subseteq f(X)\}$. Thus we have the following:

Corollary 4.7. *Let f be a monotonic function over the subset lattice generated by S. Then $\nu f = \bigcup\{X \in 2^S \mid X$ is supported for $f\}$.*

This section closes with definitions and results on support structures that we use later in this article. In what follows we fix a set S and the associated subset lattice $(2^S, \subseteq, \bigcup, \bigcap)$. The first lemma establishes that any extension of a support structure is also a support structure.

Lemma 4.8 (Extensions of Support Structures). *Let $f \in 2^S \rightarrow 2^S$ be monotonic, $(X, \prec)$ be a support structure for f, and $\prec' \subseteq S \times S$ be an extension of $\prec$. Then $(X, \prec')$ is a support structure for f.*

Proof. Follows from monotonicity of f and the fact that $\prec^{-1}(x) \subseteq (\prec')^{-1}(x)$ for all $x \in X$. □

The next lemma establishes that unions of support structures are also support structures.

Lemma 4.9 (Unions of Support Structures). *Let $f \in 2^S \rightarrow 2^S$ be monotonic, and let $\mathcal{X}$ be a set of support structures for f. Then $(S_{\mathcal{X}}, \prec_{\mathcal{X}})$ is a support structure for f, where $S_{\mathcal{X}} = \bigcup_{(S,\prec) \in \mathcal{X}} S$, and $\prec_{\mathcal{X}} = \bigcup_{(S,\prec) \in \mathcal{X}} \prec$.*

Proof. It suffices to show that for every $s \in S_{\mathcal{X}}$, $s \in f({\prec_{\mathcal{X}}}^{-1}(s))$. So fix such an s. Since $s \in S_{\mathcal{X}}$ there is $(S, \prec) \in \mathcal{X}$ such that $s \in S$, and as $(S, \prec)$ is a support structure we know that $s \in f(\prec^{-1}(s))$. That $s \in f({\prec_{\mathcal{X}}}^{-1}(s))$ is immediate from the fact that f is monotonic and $\prec^{-1}(x) \subseteq {\prec_{\mathcal{X}}}^{-1}(x)$. □

This result should be contrasted with Lemma 4.3. On the one hand, Lemma 4.9 asserts a property about all sets of support structures, whereas Lemma 4.3 only refers to sets of well-supported sets. On the other hand, Lemma 4.9 makes no guarantees about the properties of support structure $(S_{\mathcal{X}}, \prec_{\mathcal{X}})$ *vis à vis* the orderings $(S, \prec)$. In particular, if all the $(S, \prec) \in \mathcal{X}$ are well-founded, it does not follow that $\prec_{\mathcal{X}}$ is well-founded. Lemma 4.3 on the other does guarantee that a well-founded ordering over $S_{\mathcal{X}}$ does exist if each $(S, \prec) \in \mathcal{X}$ is well-founded.

Well-founded support structures can be extended to well-orderings.

Lemma 4.10 (Well-orderings for Well-supported Sets). *Let $f \in 2^S \to 2^S$ be monotonic, and let $(X, \prec)$ be a well-founded support structure for f. Then there is a well-ordering $\prec' \subseteq X \times X$ extending $\prec$ such that $(X, \prec')$ is a support structure for f.*

Proof. Follows from Lemmas 3.12 and 4.8. □

The next result is a corollary of earlier lemmas.

Corollary 4.11 (Support Structures for Fixpoints). *Let $f \in 2^S \to 2^S$ be monotonic.*

(1) $(\nu f, U_{\nu f})$ is a support structure for f.
(2) There is a well-ordering $\prec \subseteq \mu f \times \mu f$ such that $(\mu f, \prec)$ is a support structure for f.

For technical convenience in what follows we introduce the notions of *σ-compatible* and *σ-maximal* support structures for monotonic f and $\sigma \in \{\mu, \nu\}$.

Definition 4.12 (Compatible, Maximal Support Structures). Let $f \in 2^S \to 2^S$ be monotonic, let $\sigma \in \{\mu, \nu\}$, and let $(X, \prec)$ be a support structure for f.

(1) $(X, \prec)$ is *σ-compatible for f* iff either $\sigma = \nu$, or $\sigma = \mu$ and $\prec$ is well-founded.
(2) $(X, \prec)$ is *σ-maximal for f* iff $X = \sigma f$ and one of the following holds.
   (a) $\sigma = \nu$ and $\prec = U_X$.
   (b) $\sigma = \mu$ and $\prec$ is a well-ordering over $X \times X$.

Corollary 4.11 ensures that for any monotonic $f \in 2^S \to 2^S$ and $\sigma \in \{\mu, \nu\}$ there is a σ-maximal support structure for f. When $\sigma = \nu$ this σ-maximal support structure is unique, whereas this uniqueness property in general fails to hold for $\sigma = \mu$; while the fixpoint is unique, the associated well-ordering need not be.

5 THE PROPOSITIONAL MODAL MU-CALCULUS

This section defines the syntax and semantics of the modal mu-calculus and also establishes properties of the logic that will be used later in the article.

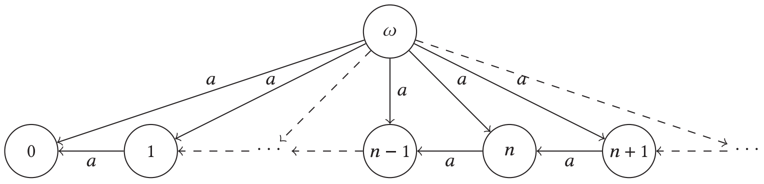
5.1 Labeled Transition Systems

LTSs are intended to model the behavior of discrete systems. Define a *sort* Σ to be the set of atomic actions that a system can perform.

Definition 5.1 (LTS). An LTS *of sort* Σ is a pair $(\mathcal{S}, \rightarrow)$, where $\mathcal{S}$ is a set of *states* and $\rightarrow \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$ is the *transition relation.* We write $s \xrightarrow{a} s'$ when $(s, a, s') \in \rightarrow$ and $s \xrightarrow{a}$ when $s \xrightarrow{a} s'$ for some $s' \in \mathcal{S}$. If $K \subseteq \Sigma$ then we write $s \xrightarrow{K} s'$ iff $s \xrightarrow{a} s'$ for some $a \in K$ and $s \xrightarrow{K}$ if $s \xrightarrow{K} s'$ for some s'. If there is no s' such that $s \xrightarrow{a} s'$ / $s \xrightarrow{K} s'$ then we denote this as $s \not\xrightarrow{a}$ / $s \not\xrightarrow{K}$.

An LTS $(\mathcal{S},\rightarrow)$ of sort Σ represents a system whose state space is $\mathcal{S}$; the presence of transition $s \xrightarrow{a} s'$ indicates that when the system is in state s, it can perform atomic action a and evolve to state s'. We now introduce two notions of *predecessors* of sets of states in an LTS.

Example 5.2. The infinite state LTS shown below is of sort $\Sigma = \{a\}$, with set of states $\mathcal{S} = \mathbb{N} \cup \{\omega\}$, and the transition relation $\rightarrow$ is such that $\omega \xrightarrow{a} n$ for all $n \in \mathbb{N}$, and $n+1 \xrightarrow{a} n$ for all $n \in \mathbb{N}$.



Definition 5.3 (Predecessor Sets). Let $(\mathcal{S},\rightarrow)$ be an LTS of sort Σ, with $S \subseteq \mathcal{S}$ and $K \subseteq \Sigma$. Then:

(1) $pred_{\langle K \rangle}(S) = \{s \in \mathcal{S} \mid \exists s' \in S \colon s \xrightarrow{K} s'\}$; and
(2) $pred_{[K]}(S) = \{s \in \mathcal{S} \mid \forall s' \in \mathcal{S} \colon s \xrightarrow{K} s' \implies s' \in S\}$.

If state $s \in pred_{\langle K \rangle}(S)$ then it has at least one outgoing K-transition leading to a state in S, while $s \in pred_{[K]}(S)$ holds iff every outgoing K-transition from s leads to S. Note that if $s \overset{K}{\not\rightarrow}$ then $s \in pred_{[K]}(S)$ but $s \notin pred_{\langle K \rangle}(S)$. It immediately follows from the definitions that the operators satisfy the following properties.

Lemma 5.4. *Let $(\mathcal{S},\rightarrow)$ be an LTS of sort Σ, with $K \subseteq \Sigma$ and $S_1, S_2 \subseteq \mathcal{S}$. If $S_1 \subseteq S_2$ then $pred_{[K]}(S_1) \subseteq pred_{[K]}(S_2)$ and $pred_{\langle K \rangle}(S_1) \subseteq pred_{\langle K \rangle}(S_2)$.*

5.2 Propositional Modal Mu-calculus

The propositional modal mu-calculus, which we usually just call the mu-calculus, is a logic for describing properties of states in LTSs. The version of the logic considered here matches the one in [11], which slightly extends [44] by allowing sets of labels in the modalities. We first define the set of formulas of the mu-calculus, then the *well-formed* formulas. The latter will be the object of study in this article.

Definition 5.5 (Mu-calculus Formulas). Let Σ be a sort and Var a countably infinite set of *propositional variables.* Then formulas of the propositional modal mu-calculus over Σ and Var are given by the following grammar, where $K \subseteq \Sigma$ and $Z \in \text{Var}$.

$$\Phi ::= Z \mid \neg\Phi' \mid \Phi_1 \wedge \Phi_2 \mid [K]\Phi' \mid \nu Z.\Phi'$$

We assume the usual definitions of subformula, and so forth. To define the well-formed mu-calculus formulas, we first review the notions of free, bound and positive variables. Occurrences of Z in $\nu Z.\Phi'$ are said to be *bound*; an occurrence of a variable in a formula that is not bound within the formula is called *free.* A variable Z is free within a formula if it has at least one free occurrence in the formula, and is *positive* in Φ if every free occurrence of Z in Φ occurs inside the scope of an even number of negations. We now define the well-formed mu-calculus formulas as follows:

Definition 5.6 (Well-formed Mu-calculus Formulas). A mu-calculus formula over Σ and Var is *well-formed* if each of its subformulas of form $\nu Z.\Phi$ satisfies: Z is positive in Φ. We use $\mathbb{F}^{\Sigma}_{\text{Var}}$ for the set of well-formed mu-calculus formulas over Σ and Var.

We denote substitution for free variables in the usual fashion: if $Z_1 \cdots Z_n \in \text{Var}^*$ is duplicate-free and $\Phi_1 \cdots \Phi_n \in \mathbb{F}^*$ then we write $\Phi[Z_1 \cdots Z_n := \Phi_1 \cdots \Phi_n]$ for the simultaneous capture-free substitution of each Z_i by Φ_i in Φ. We also use the following standard derived operators.

$$\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2) \quad \langle K\rangle\Phi = \neg[K]\neg\Phi \quad \mu Z.\Phi = \neg\nu Z.\neg\Phi[Z := \neg Z] \quad \mathsf{tt} = \nu Z.Z \quad \mathsf{ff} = \neg\mathsf{tt}$$

In the definition of $\mu Z.\Phi$, note that if Z is positive in Φ then it is also positive in $\neg\Phi[Z := \neg Z]$. Following standard convention, we refer to $\wedge$ and $\vee$, $[K]$ and $\langle K\rangle$, and ν and μ as *duals*. Formulas extended with these dual operators are in *positive normal form* iff all negation symbols directly apply to free variable occurrences. It is well-known that every well-formed formula can be rewritten to positive normal form when the duals are included in the logic. We refer to formulas of form $\nu Z.\Phi$ or $\mu Z.\Phi$ as *fixpoint formulas* and write $\sigma Z.\Phi$ for a generic such formula (so σ may be either ν or μ).

Example 5.7. The following are examples of mu-calculus formulas.

(1) $\mu X.[a]X$ expresses that all paths consisting of only a-actions are finite, in other words, inevitably, either at some point an action other than a must be done, or no action is possible.
(2) $\nu X.\mu Y.(\langle a\rangle X \vee \langle\Sigma \setminus \{a\}\rangle X)$ expresses there is a path along which a happens infinitely often.
(3) $\mu X.\nu Y.([a]Y \wedge [\Sigma \setminus \{a\}]X)$ expresses that all infinite paths eventually end with only a-actions.

We use *valuations* to handle propositional variables in the semantics of mu-calculus formulas.

Definition 5.8 (Valuations). Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be an LTS and Var a countably infinite set of variables. Then a *valuation for* Var *over* $\mathcal{T}$ is a function $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$.

Since a valuation $\mathcal{V}$ is a function, standard operations on functions such as $\mathcal{V}[Z_1 \cdots Z_n := S_1 \cdots S_n]$, where $Z_1 \cdots Z_n \in \text{Var}^*$ is duplicate-free and $S_1 \cdots S_n \in (2^{\mathcal{S}})^*$, are applicable.

The semantics of the mu-calculus is now defined as follows:

Definition 5.9 (Mu-calculus Semantics). Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be an LTS of sort Σ, Var a countably infinite set of free variables, and $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ a valuation. Then the semantic function $||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}} \subseteq \mathcal{S}$, where $\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$, is defined as follows:

$$||\,Z\,||^{\mathcal{T}}_{\mathcal{V}} = \mathcal{V}(Z), \qquad ||\,\neg\Phi\,||^{\mathcal{T}}_{\mathcal{V}} = \mathcal{S} \setminus ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}},$$
$$||\,\Phi_1 \wedge \Phi_2\,||^{\mathcal{T}}_{\mathcal{V}} = ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}} \cap ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}, \qquad ||\,[K]\Phi\,||^{\mathcal{T}}_{\mathcal{V}} = \mathit{pred}_{[K]}\left(||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}\right),$$
$$||\,\nu Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}} = \bigcup\{S \subseteq \mathcal{S} \mid S \subseteq ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}[Z:=S]}\}.$$

If $s \in ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$ then we say that s *satisfies* Φ in the context of $\mathcal{T}$ and $\mathcal{V}$.

For the dual operators one may derive the following semantic equivalences.

$$||\,\Phi_1 \vee \Phi_2\,||^{\mathcal{T}}_{\mathcal{V}} = ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}} \cup ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}, \qquad ||\,\langle K\rangle\Phi\,||^{\mathcal{T}}_{\mathcal{V}} = \mathit{pred}_{\langle K\rangle}\left(||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}\right),$$
$$||\,\mu Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}} = \bigcap\{S \subseteq \mathcal{S} \mid ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}[Z:=S]} \subseteq S\}.$$

Example 5.10. Consider the LTS from Example 5.2. All states in this LTS satisfy the mu-calculus formula $\mu X.[a]X$ (Example 5.7 (1)), since every a-path is finite. Note that this is the case even though ω has (countably) infinitely many outgoing a-transitions, since once an a-transition to a particular state n is taken from ω, it only takes n transitions to end up in state 0, from which no a-transition is possible.

We frequently wish to view formulas as functions of their free variables. The next definition introduces this concept at both the syntactic and semantic level.

Definition 5.11 (Formula Functions). Let $Z \in \text{Var}$ be a variable and $\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$ be a formula.

(1) The *syntactic function*, $Z.\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}} \rightarrow \mathbb{F}^{\Sigma}_{\text{Var}}$, for Z and Φ is defined as
$$(Z.\Phi)(\Phi') = \Phi[Z := \Phi'].$$

(2) Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be an LTS of sort Σ, and $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ a valuation over $\mathcal{T}$. Then the *semantic function*, $||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}} \in 2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$, for Z and Φ is defined as

$$||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}(S) = ||\, \Phi\, ||_{\mathcal{V}[Z:=S]}^{\mathcal{T}}.$$

We now state a well-known monotonicity result for formulas in which Z is positive.

Lemma 5.12 (Mu-calculus Monotonicity). *Fix $\mathcal{T} = (\mathcal{S},\rightarrow)$ and $\mathcal{V}$, and let $\Phi \in \mathbb{F}_{\text{Var}}^{\Sigma}$ be such that $Z \in \text{Var}$ is positive in Φ. Then $||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}} \in 2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$ is monotonic over the subset lattice for $\mathcal{S}$.*

It turns out that the semantics of well-formed formulas $\nu Z.\Phi$ and $\mu Z.\Phi$ with respect to $\mathcal{T} = (\mathcal{S},\rightarrow)$ can be characterized as the greatest and least fixpoints, respectively, of $||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}$ over the subset lattice induced by $\mathcal{S}$. In particular, Lemma 5.12 guarantees the monotonicity of $||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}$ over this lattice; Lemma 3.16 then implies the characterization. The next lemma formalizes this insight.

Lemma 5.13 (Fixpoint Characterizations of Formula Functions). *Fix $\mathcal{T}$ and $\mathcal{V}$, let $Z \in \text{Var}$, and let $\Phi \in \mathbb{F}_{\text{Var}}^{\Sigma}$ be such that $Z \in \text{Var}$ is positive in Φ. Then $\nu\left(||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}\right) = ||\, \nu Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}$ and $\mu\left(||\, Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}\right) = ||\, \mu Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}}$.*

We now give some identities on mu-calculus formulas that we will use later in this article. The first result establishes a correspondence between substitution and valuation updates.

Lemma 5.14 (Substitution and Valuations). *Fix $\mathcal{T}$ and $\mathcal{V}$, let $\Phi, \Phi_1, \ldots, \Phi_n \in \mathbb{F}_{\text{Var}}^{\Sigma}$ and let $Z_1 \cdots Z_n \in \text{Var}^*$ be duplicate-free. Then*

$$||\, \Phi[Z_1 \cdots Z_n := \Phi_1 \cdots \Phi_n]\, ||_{\mathcal{V}}^{\mathcal{T}} = ||\, \Phi\, ||_{\mathcal{V}[Z_1 \cdots Z_n := ||\, \Phi_1\, ||_{\mathcal{V}}^{\mathcal{T}} \cdots ||\, \Phi_n\, ||_{\mathcal{V}}^{\mathcal{T}}]}^{\mathcal{T}}.$$

Lemmas 5.13 and 5.14 guarantee that formulas can be *unfolded.*

Lemma 5.15 (Fixpoint Unfolding). *Fix $\mathcal{T}$ and $\mathcal{V}$, and let $\sigma Z.\Phi$ be a fixpoint formula. Then $||\, \sigma Z.\Phi\, ||_{\mathcal{V}}^{\mathcal{T}} = ||\, \Phi[Z := \sigma Z.\Phi]\, ||_{\mathcal{V}}^{\mathcal{T}}$.*

6 BASE PROOF SYSTEM

This section defines the base proof system for the mu-calculus considered in this article. It mirrors the ones given in [11, 13] and is intended to prove that sets of states in a transition system satisfy mu-calculus formulas. Later in the article, we will extend this proof system in various ways. In what follows, fix sort Σ and countably infinite propositional variable set Var.

6.1 Definition Lists and Sequents

The proof system reasons about *sequents*, which make statements about sets of states satisfying mu-formulas. Our sequents also involve *definition lists*, which are used in the construction of proofs to control the unfolding of fixpoint formulas. We define definition lists and sequents below.

Definition lists. Definition lists bind fresh variables in Var to formulas. In a proof setting, such a list records the fixpoint formulas that have been unfolded previously, so that decisions about whether or to unfold again later in the proof can be made. Here we define definition lists formally and establish basic results about them.

Definition 6.1 (Definition Lists). A *definition list* Δ is a finite sequence $(U_1 = \Phi_1) \cdots (U_n = \Phi_n)$, with each $U_i \in \text{Var}$ and $\Phi_i \in \mathbb{F}_{\text{Var}}^{\Sigma}$, satisfying the following:

(1) If $i \neq j$ then $U_i \neq U_j$.
(2) For all $1 \leq i, j \leq n$, U_i cannot appear bound anywhere in Φ_j.
(3) If $i \leq j$ then U_j cannot appear free in Φ_i.

The individual $(U_i = \Phi_i)$ in a definition list are sometimes called *definitions*, with each U_i referred to as a *definitional constant*. We also define $\Delta(U_i) = \Phi_i$ to be the formula associated with U_i in Δ and $\text{dom}(\Delta) = \{U_1, \ldots, U_n\}$ to be the set of definitional constants in Δ.

A definition list consists of a sequence of bindings, or definitions, of form $(U_i = \Phi_i)$. The constraints ensure that every U_i is unique and not part of any σ operator inside Φ_i. U_i may appear free in definitions to the right of $(U_i = \Phi_i)$, but not in Φ_i or in definitions to the left. Since definition lists are sequences, the sequence notations defined in Section 3.1, including ε, $\cdot$ and $\preceq$, are applicable.

Syntactically, definition lists give rise to a notion of *iterated substitution*.

Definition 6.2 (Formula Expansion by a Definition List). Let Δ be a definition list and $\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$. Then $\Phi[\Delta]$, the *expansion* of Φ with respect to Δ, is defined inductively on Δ as follows:

— $\Phi[\varepsilon] = \Phi$.
— $\Phi[\Delta \cdot (U = \Psi)] = (\Phi[U := \Psi])\,[\Delta]$.

Note that $\Phi[\Delta]$ contains no occurrences of any elements of $\text{dom}(\Delta)$.

In a similar vein, we may define the semantic extension, $\mathcal{V}[\Delta]$, of valuation $\mathcal{V}$ by definition list Δ. Essentially, $\mathcal{V}[\Delta]$ updates $\mathcal{V}$ with the semantic interpretation of the equations appearing in Δ.

Definition 6.3 (Valuation Extension by a Definition List). Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be an LTS over Σ, $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ a valuation, and Δ a definition list. Then $\mathcal{V}[\Delta]$, the *extension* of $\mathcal{V}$ by Δ, is defined inductively on Δ as follows:

(1) $\mathcal{V}[\varepsilon] = \mathcal{V}$.
(2) $\mathcal{V}[(U = \Phi) \cdot \Delta] = \left(\mathcal{V}\,[\,U := ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}\,]\right)[\Delta]$.

Lemma 6.4 establishes a correspondence between $||\,\Phi[\Delta]\,||^{\mathcal{T}}_{\mathcal{V}}$ and $||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]}$.

Lemma 6.4 (Definition-list Correspondence). *Let $\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$. Then for every LTS $\mathcal{T}$ over Σ, definition list Δ, and valuation $\mathcal{V}$, $||\,\Phi[\Delta]\,||^{\mathcal{T}}_{\mathcal{V}} = ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]}$.*

Proof. Fix arbitrary Φ and LTS $\mathcal{T}$ over Σ. The proof proceeds by induction on Δ, and uses the observation that, for non-empty definition list $\Delta = (U_1 = \Phi_1) \cdot \Delta'$ and mu-formulas Φ, we have $\Phi[\Delta] = (\Phi[\Delta'])\ [U_1 := \Phi_1]$. □

We close our treatment of definition lists by showing the following useful fact about unfolding when $\Delta(U) = \sigma Z.\Phi$ for some Φ.

Lemma 6.5 (Definitional-constant Unfolding). *Let $\mathcal{T}$ be an LTS over Σ, Δ be a definition list with $\Delta(U) = \sigma Z.\Phi$, and $\mathcal{V}$ be a valuation. Then $||\,U\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]} = ||\,\Phi[Z := U]\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]}$.*

Proof. Follows from Lemma 5.15 and the fact that $||\,U\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]} = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}[\Delta]}$. □

Sequents. We can now define the sequents used in the base proof system.

Definition 6.6 (Sequents). Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be an LTS over Σ and $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ a valuation over Var. Then a *sequent over $\mathcal{T}$ and $\mathcal{V}$* has form $S \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi$, where $S \subseteq \mathcal{S}$, Δ is a definition list, and $\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$ has the following properties.

(1) Φ is in positive normal form.
(2) Every $U \in \text{dom}(\Delta)$ is positive in Φ.
(3) No $U \in \text{dom}(\Delta)$ is bound in Φ.

$$\wedge\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_1 \wedge \Phi_2}{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_1 \quad S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_2} \qquad \vee\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_1 \vee \Phi_2}{S_1 \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_1 \quad S_2 \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi_2}\ S = S_1 \cup S_2$$

$$[K]\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} [K]\Phi}{S' \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi}\ S' = \{s' \in \mathcal{S} \mid \exists s \in S. s \xrightarrow{K} s'\}$$

$$\langle K\rangle\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \langle K\rangle\Phi}{f(S) \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi}\ f \in S \to \mathcal{S} \text{ is such that } \forall s \in S\colon s \xrightarrow{K} f(s)$$

$$\sigma Z\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi}{S \vdash_{\Delta\cdot(U=\sigma Z.\Phi)}^{\mathcal{T},\mathcal{V}} U}\ U \text{ fresh}^4 \qquad \text{Un}\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} U}{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi[Z := U]}\ \Delta(U) = \sigma Z.\Phi$$

$$\text{Thin}\ \frac{S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi}{S' \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi'}\ S \subseteq S'$$

Fig. 1. Proof rules for mu-calculus sequents.

We use $\mathbf{S}_{\mathcal{V}}^{\mathcal{T}}$ for the set of sequents over $\mathcal{T}$ and $\mathcal{V}$. If $\mathbf{s} = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ is in $\mathbf{S}_{\mathcal{V}}^{\mathcal{T}}$, then we access the components of $\mathbf{s}$ as follows: $st(\mathbf{s}) = S$, $dl(\mathbf{s}) = \Delta$, and $fm(\mathbf{s}) = \Phi$.

The intended interpretation of sequent $S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ is that it is valid iff every $s \in S$ satisfies Φ, where $\mathcal{V}$ is used to interpret free propositional variables $Z \notin \text{dom}(\Delta)$ that appear in Φ and Δ is used for definitional constants $U \in \text{dom}(\Delta)$ that occur in Φ. This notion is formalized as follows:

Definition 6.7 (Sequent Semantics and Validity). Let $\mathbf{s} = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ be a sequent in $\mathbf{S}_{\mathcal{V}}^{\mathcal{T}}$.

(1) The *semantics* of $\mathbf{s}$ is defined as $\|\mathbf{s}\| = \|\Phi\|_{\mathcal{V}[\Delta]}^{\mathcal{T}}$.
(2) Sequent $\mathbf{s}$ is *valid* iff $S \subseteq \|\mathbf{s}\|$.

6.2 Proof Rules and Tableaux

We now present the proof rules used in this article for the mu-calculus formula sequents described in the previous section. These proof rules come from [11, 13] and resemble traditional natural-deduction-style proof rules. Following [11, 13]; however, we write the conclusion of the proof rule above the premises to emphasize that the conclusion is a "goal" and the premises are "subgoals" in a goal-directed proof-search strategy. In what follows we fix sort Σ, transition system $\mathcal{T} = (\mathcal{S}, \to)$ over Σ, variable set Var and valuation $\mathcal{V} \in \text{Var} \to 2^{\mathcal{S}}$.

Definition 6.8 (Proof Rule). A *proof rule* has form

$$\textit{name}\ \frac{\mathbf{s}}{\mathbf{s}_1 \ \cdots\ \mathbf{s}_n}\ \textit{side condition}$$

where *name* is the name of the rule, $\mathbf{s}_1 \cdots \mathbf{s}_n \in (\mathbf{S}_{\mathcal{V}}^{\mathcal{T}})^*$ is a sequence of sequents called the *premises* of the rule, $\mathbf{s} \in \mathbf{S}_{\mathcal{V}}^{\mathcal{T}}$ is the *conclusion* of the rule, and the optional *side condition* is a property determining when the rule may be applied.

Figure 1 gives the proof rules from [13], lightly adapted to conform to our notational conventions. The rules are named for the top-level operators appearing in the formulas of the sequents to which

[4]Here U is fresh if it has not been previously used anywhere in a proof currently being constructed using these proof rules.

they apply. Rule σZ is actually short-hand for two rules—one for each possible value, μ and ν, of σ—and asserts that a definition involving fresh constant U is added to the end of the definition list of the subgoal. The Thin rule is needed to ensure completeness of the proof system.

The side condition of Rule $\langle K \rangle$ refers to a function f that is responsible for selecting, for each state s in the goal sequent, a *witness* state $f(s)$ such that $s \xrightarrow{K} f(s)$ and $f(s)$ is in the subgoal sequent. To apply this rule, a specific function f must be identified that computes these witness states. We call the function associated to an application of $\langle K \rangle$ the *witness function* of the application. We further define the set of *rule applications* for a given LTS $(\mathcal{S}, \rightarrow)$ over Σ as follows:

$$\text{RAppl} = \{\wedge, \vee, [K], \mu Z, \nu Z, \text{Un}, \text{Thin}\} \cup \{(\langle K \rangle, f) \mid f \text{ is a witness function}\}.$$

Note that rule applications are either rule names, if the rule being applied is not $\langle K \rangle$, or pairs of form $(\langle K \rangle, f)$ where f is the witness function used in applying the $\langle K \rangle$ rule. If a is a rule application then we write $rn(a)$ for the rule name used in a. Formally, $rn(a)$ is defined as follows:

$$rn(a) = \begin{cases} \langle K \rangle & \text{if } a = (\langle K \rangle, f) \\ a & \text{otherwise.} \end{cases}$$

Intuitively, proofs are constructed as follows. Suppose $S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ is a sequent we wish to prove. Based on the form of Φ we select a rule whose conclusion matches this sequent and apply it, generating the corresponding premises and witness function as required by the rule. We then recursively build proofs for these premises. The proof construction process terminates when the validity (or lack thereof) of the current sequent can be immediately established. The resulting proofs can be viewed as trees, called *tableaux* in [11, 13], and which we formalize as follows:

Definition 6.9 ((Partial) Tableaux).

(1) A *partial tableau* has form $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, where:
- $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ is a finite non-empty ordered tree (cf. Definition 3.18).
- Partial function $\rho \in \mathbf{N} \rightarrow_{\perp} \text{RAppl}$ satisfies: $\rho(\mathbf{n})\perp$ iff $\mathbf{n}$ is a leaf of $\mathbf{T}$.
- $\mathcal{T} = (\mathcal{S}, \rightarrow)$ is an LTS.
- $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ is a valuation.
- Function $\lambda \in \mathbf{N} \rightarrow \mathbf{S}_{\mathcal{V}}^{\mathcal{T}}$, the *sequent labeling*, satisfies: if $\rho(\mathbf{n}) \in \text{RAppl}$ then $\rho(\mathbf{n})$, $\lambda(\mathbf{n})$ and $\lambda(cs(\mathbf{n}))$[5] satisfy the form and side condition associated with rule $rn(\rho(\mathbf{n}))$.

Elements of $\mathbf{N}$ are sometimes called the *proof nodes* of $\mathbb{T}$.

(2) Partial tableau $(\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ is a *complete tableau*, or simply a *tableau*, iff $dl(\lambda(\mathbf{r})) = \varepsilon$ (i.e., the definition list in the root is empty), and all leaves $\mathbf{n}$ in $\mathbf{T}$ are *terminal*, i.e., satisfy one of the following:
  (a) $fm(\lambda(\mathbf{n})) = Z$ or $fm(\lambda(\mathbf{n})) = \neg Z$ for some $Z \in \text{Var} \setminus \text{dom}(dl(\lambda(\mathbf{n})))$ (in this case $\mathbf{n}$ is called a *free leaf*); or
  (b) $fm(\lambda(\mathbf{n})) = \langle K \rangle \Phi$ for some Φ and there is $s \in st(\lambda(\mathbf{n}))$ such that $s \stackrel{K}{\not\rightarrow}$ (in this case $\mathbf{n}$ is called a *diamond leaf*); or
  (c) $fm(\lambda(\mathbf{n})) = U$ for some $U \in \text{dom}(dl(\lambda(\mathbf{n})))$, and there is $\mathbf{m} \in A{\uparrow}(\mathbf{n})$[6] such that $fm(\lambda(\mathbf{m})) = U$ and $st(\lambda(\mathbf{n})) \subseteq st(\lambda(\mathbf{m}))$ (in this case $\mathbf{n}$ is called a *σ-leaf*).

We sometimes refer to a σ-leaf as a μ- / ν-leaf if $\Delta(U) = \mu \ldots$ / $\Delta(U) = \nu \ldots$. The deepest node $\mathbf{m}$ making $\mathbf{n}$ a σ-leaf is the *companion node* of $\mathbf{n}$; we also say $\mathbf{n}$ is a *companion leaf* of $\mathbf{m}$.

[5]Recall that $cs(\mathbf{n})$ is the sequence of the children of node $\mathbf{n}$ in left-to-right order.
[6]Recall that $A{\uparrow}(\mathbf{n})$ is the set of strict ancestors of $\mathbf{n}$.

A σ-leaf in a tableau has a unique companion node based on the Definition 6.9(2), but a node may be a companion node for multiple (or no) companion leaves; all such companion leaves must be in the subtree rooted at the node, however. The definition of terminal leaf in [11, 13] also includes an extra case, $st(\lambda(\mathbf{n})) = \emptyset$, in addition to the Conditions 2(a)–2(c) in our definition above. It turns out that this case is unnecessary, as the completeness results in Section 8 show.

In what follows we adopt the following notational shorthands for proof nodes in partial tableaux.

Notation 6.10 (Proof Nodes). Let $\mathbf{n}$ be a proof node in partial tableau $(\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$.

- $\mathbf{n} = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ means $\lambda(\mathbf{n}) = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$.
- $\|\, \mathbf{n} \,\| = \|\, \lambda(\mathbf{n}) \,\| = \|\, fm(\lambda(\mathbf{n})) \,\|_{\mathcal{V}}^{\mathcal{T}}$.
- $\mathbf{n}$ is valid iff $\lambda(\mathbf{n})$ is valid.
- $st(\mathbf{n}) = st(\lambda(\mathbf{n}))$.
- $dl(\mathbf{n}) = dl(\lambda(\mathbf{n}))$.
- $fm(\mathbf{n}) = fm(\lambda(\mathbf{n}))$.

Companion nodes and leaves feature prominently later, so we introduce the following notation and remark on a semantic property that they satisfy.

Notation 6.11 (Companion Nodes and Leaves). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a partial tableau, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$.

(1) The set $\mathbf{C}_{\mathbb{T}} \subseteq \mathbf{N}$ of companion nodes of $\mathbb{T}$ is given by: $\mathbf{C}_{\mathbb{T}} = \{\mathbf{n} \in \mathbf{N} \mid \rho(\mathbf{n}) = \text{Un}\}$.
(2) Let $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$. Then the set $\mathbf{CL}_{\mathbb{T}}(\mathbf{n}) \subseteq D{\downarrow}(\mathbf{n})$ of companion leaves of $\mathbf{n}$ is given as follows, where $\mathbf{C}_{\mathbb{T},\mathbf{n}} = \mathbf{C}_{\mathbb{T}} \cap D{\downarrow}(\mathbf{n})$ are the companion nodes of $\mathbb{T}$ that are strict descendants of $\mathbf{n}$.

$$\mathbf{CL}_{\mathbb{T}}(\mathbf{n}) = \{\mathbf{n}' \in D{\downarrow}(\mathbf{n}) \mid c(\mathbf{n}') = \emptyset \wedge fm(\mathbf{n}') = fm(\mathbf{n}) \wedge st(\mathbf{n}') \subseteq st(\mathbf{n})\} \setminus \bigcup_{\mathbf{n}' \in \mathbf{C}_{\mathbb{T},\mathbf{n}}} \mathbf{CL}_{\mathbb{T}}(\mathbf{n}').$$

Note that if $\mathbf{n}$ is a companion node then it must be the case that $fm(\mathbf{n}) = U$ for some definitional constant U defined in the definition list $dl(\mathbf{n})$ of $\mathbf{n}$. Given a companion node $\mathbf{n}$ in $\mathbb{T}$ its associated companion leaves, $\mathbf{CL}_{\mathbb{T}}(\mathbf{n})$, consist of nodes $\mathbf{n}'$ that: (i) are leaves ($c(\mathbf{n}') = \emptyset$); (ii) have the same definitional constant as $\mathbf{n}$ for their formula ($fm(\mathbf{n}') = fm(\mathbf{n})$); (iii) have their state sets included in the state set of $\mathbf{n}$ ($st(\mathbf{n}') \subseteq st(\mathbf{n})$); and (iv) are not companion leaves of any companion node that is a strict descendant of $\mathbf{n}$ ($\bigcup_{\mathbf{n}' \in \mathbf{C}_{\mathbb{T},\mathbf{n}}} \mathbf{CL}_{\mathbb{T}}(\mathbf{n})$).

Lemma 6.12 (Semantic Invariance of Definitional Constants). *Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a partial tableau. Then for any proof nodes $\mathbf{n}$ and $\mathbf{n}'$ in $\mathbb{T}$ and definitional constant U, if $fm(\mathbf{n}) = fm(\mathbf{n}') = U$ then $\|\, \mathbf{n} \,\| = \|\, \mathbf{n}' \,\|$.*

Proof. Follows from the fact that Rule σZ, which is the only rule that modifies definition lists, introduces a given definitional constant at most once in a partial tableau, and at the end of the associated definition list. This, plus the fact that $\|\, U \,\|_{\mathcal{V}[\Delta]}^{\mathcal{T}}$ is only influenced by the prefix of Δ up to and including the definition of U, gives the desired result. □

A tableau is a candidate proof; while the structure of a tableau ensures the correct application of proof rules, this alone does not guarantee that a tableau is a valid proof. The notion of *successful tableau* fills this gap.

Definition 6.13 (Successful Leaf/Tableau). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a tableau. Then leaf node $\mathbf{n}$ in $\mathbb{T}$ is *successful* iff one of the following holds.

(1) $fm(\mathbf{n}) = Z$, where $Z \in \text{Var} \setminus \text{dom}(dl(\mathbf{n}))$, and $S \subseteq \mathcal{V}(Z)$; or
(2) $fm(\mathbf{n}) = \neg Z$, where $Z \in \text{Var} \setminus \text{dom}(dl(\mathbf{n})))$, and $S \cap \mathcal{V}(Z) = \emptyset$; or

(3) $\mathbf{n}$ is a ν-leaf; or
(4) $\mathbf{n}$ is a μ-leaf satisfying the condition given below in Definition 6.17.

A tableau is successful iff all its leaves are successful.

The rest of this section is devoted to defining the success of μ-leaves, which is more complicated than the other cases and spans several definitions. Intuitively, the success of a μ-leaf depends on the well-foundedness of an *extended dependency ordering* involving the companion node of the leaf. Our definition of this ordering closely matches similar ones in [10, 13] and is given in three stages. In what follows, fix partial tableau $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, with $\mathbf{T} = (\mathbf{N}, r, p, cs)$.

The *local dependency ordering* captures the one-step dependencies between a state in a node in $\mathbb{T}$ and states in the node's children.

Definition 6.14 (Local Dependency Ordering). Let $\mathbf{n}, \mathbf{n}' \in \mathbf{N}$ be proof nodes in $\mathbb{T}$, with $\mathbf{n}' \in c(\mathbf{n})$ a child of $\mathbf{n}$. Then $s' <_{\mathbf{n}',\mathbf{n}} s$ iff $s' \in st(\mathbf{n}')$, $s \in st(\mathbf{n})$, and one of the following hold:

(1) $\rho(\mathbf{n}) = [K]$ and $s \xrightarrow{K} s'$; or
(2) $\rho(\mathbf{n}) = (\langle K \rangle, f)$ and $s' = f(s)$; or
(3) $rn(\rho(\mathbf{n})) \notin \{[K], \langle K \rangle\}$ and $s = s'$.

Note that since $\mathbf{n}' \in c(\mathbf{n})$, node $\mathbf{n}$ is internal and $\rho(\mathbf{n})$ is defined.

We next extend the local dependency ordering to capture dependencies between states in a node and states in descendants of the node.

Definition 6.15 (Dependency Ordering). Let $\mathbf{n}, \mathbf{n}'$ be proof nodes in $\mathbb{T}$. Then $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$ iff $s' \in st(\mathbf{n}')$, $s \in st(\mathbf{n})$, and one of the following holds.

(1) $\mathbf{n} = \mathbf{n}'$ and $s = s'$, or
(2) There exist proof node $\mathbf{m}$ and $s'' \in st(\mathbf{m})$ with $s' \lessdot_{\mathbf{n}',\mathbf{m}} s''$ and $s'' <_{\mathbf{m},\mathbf{n}} s$.

The definition of $\lessdot_{\mathbf{n}',\mathbf{n}}$ is inductive, and may be seen as analogous to the transitive and reflexive closure of $<_{\mathbf{n}',\mathbf{n}}$, modulo the node indices $\mathbf{n}'$ and $\mathbf{n}$ decorating $\lessdot_{\mathbf{n}',\mathbf{n}}$. It is easy to see that if $s' <_{\mathbf{n}',\mathbf{n}} s$ then $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$, and that if $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$ then $\mathbf{n}' \in D(\mathbf{n})$.[7]

In the third and final of our definitions, we allow *cycling* through states in companion nodes that are descendants of a given node.

Definition 6.16 (Extended Dependency Ordering). The *extended dependency ordering*, $<:_{\mathbf{n}',\mathbf{n}}$, and the *companion-node ordering*, $<:_{\mathbf{m}}$, are defined mutually recursively as follows:

(1) Let $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$ be a companion node, with $s, s' \in st(\mathbf{m})$. Then $s' <:_{\mathbf{m}} s$ iff there is a companion leaf $\mathbf{m}' \in \mathbf{CL}_{\mathbb{T}}(\mathbf{m})$ with $s' \in st(\mathbf{m}')$ and $s' <:_{\mathbf{m}',\mathbf{m}} s$.
(2) Let $\mathbf{n}, \mathbf{n}'$ be proof nodes in $\mathbb{T}$. Then $s' <:_{\mathbf{n}',\mathbf{n}} s$ iff $s' \in st(\mathbf{n}')$, $s \in st(\mathbf{n})$ and one of the following holds.
    (a) $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$; or
    (b) there exists $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$, with $\mathbf{m} \neq \mathbf{n}$ and $\mathbf{m} \neq \mathbf{n}'$, and $t, t' \in st(\mathbf{m})$, such that: $s' <:_{\mathbf{n}',\mathbf{m}} t'$, $t' <:_{\mathbf{m}}^{+} t$,[8] and $t \lessdot_{\mathbf{m},\mathbf{n}} s$.

Intuitively, $s' <:_{\mathbf{n}',\mathbf{n}} s$ captures the following semantic dependency: $s' <:_{\mathbf{n}',\mathbf{n}} s$ if the proof that s is in the semantics of $fm(\mathbf{n})$ depends on the fact that s' is in the semantics of $fm(\mathbf{n}')$. If $\mathbf{n}'$ is a companion leaf of $\mathbf{n}$, this means that in order to prove that s is in the fixed point of the formula

[7]Recall that $D(\mathbf{n})$ are descendants of $\mathbf{n}$; see Section 3.5.
[8]Recall that if R is a binary relation then R^+ is the transitive closure of R.

associated with $fm(\mathbf{n})$, it is also required that s is in the fixed point of $fm(\mathbf{n})$ (note that since $\mathbf{n}'$ is a companion leaf of $\mathbf{n}$ it must be the case that $fm(\mathbf{n}') = fm(\mathbf{n})$). Recall that the states in a companion leaf are also in the leaf's companion node; consequently, if there is a dependency involving a state in the companion node and another state in one of the node's companion leaves. This is used in the first part of the definition to define the ordering $<:_{\mathbf{m}}$ on companion nodes.

For the ordering $<:_{\mathbf{n}',\mathbf{n}}$, the first case in the definition indicates that this relation holds if applying a sequence of proof rules starting at $\mathbf{n}$ leads to $\mathbf{n}'$, with the rules $[K]$ and $\langle K\rangle$ inducing state changes in the dependency relation as the rules are applied. In the second case, the dependency chain can cycle through a companion node if there is a dependency chain from the companion node to one of its companion leaves (this is captured in the relation $<:^{+}_{\mathbf{m}}$). The dependency can be extended with a dependency involving the second state, but starting from the companion node. This explains the appearance of the transitive-closure operation in this case.

The success criterion for μ-leaves (which is really a condition on the companion nodes for these leaves) in a complete tableau can now be given as follows.

Definition 6.17 (Successful μ-leaf). Let $\mathbf{n}'$ be a μ-leaf in tableau $\mathbb{T}$ and let $\mathbf{n}$ be the companion node of $\mathbf{n}'$. Then $\mathbf{n}'$ is successful if and only if $<:_{\mathbf{n}}$ is well-founded.

Note that this definition implies that either all μ-leaves having the same companion node are successful, or none are.

Example 6.18. We now revisit our running example, with the infinite-state LTS $\mathcal{T}$, from Example 5.2, and the mu-calculus formula $\mu X.[a]X$. We prove that all states in the LTS satisfy the formula using the proof system. Fix some arbitrary valuation $\mathcal{V}$ (since the formula does not contain any free variables, this valuation is not used in the proof).

$$
\mu Z\,\dfrac{\mathbf{n}_0 = \mathbb{N}\cup\{\omega\} \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \mu X.[a]X}{\text{Un}\,\dfrac{\mathbf{n}_1 = \mathbb{N}\cup\{\omega\} \vdash^{\mathcal{T},\mathcal{V}}_{(U=\mu X.[a]X)} U}{[a]\,\dfrac{\mathbf{n}_2 = \mathbb{N}\cup\{\omega\} \vdash^{\mathcal{T},\mathcal{V}}_{(U=\mu X.[a]X)} [a]U}{\mathbf{n}_3 = \mathbb{N} \vdash^{\mathcal{T},\mathcal{V}}_{(U=\mu X.[a]X)} U}}}
$$

Note that $\mathbf{n}_3$ is a leaf-node in the tableau, with companion node $\mathbf{n}_1$. The local dependency orderings are as follows: $s <_{\mathbf{n}_1,\mathbf{n}_0} s$ and $s <_{\mathbf{n}_2,\mathbf{n}_1} s$ for all $s \in \mathbb{N}\cup\{\omega\}$; $s <_{\mathbf{n}_3,\mathbf{n}_2} \omega$ and $s <_{\mathbf{n}_3,\mathbf{n}_2} (s+1)$ for all $s \in \mathbb{N}$. This results in dependency ordering $s \lessdot_{\mathbf{n}_3,\mathbf{n}_1} \omega$ and $s \lessdot_{\mathbf{n}_3,\mathbf{n}_1} (s+1)$ for all $s \in \mathbb{N}$. Since there are no nested companion nodes, $<:_{\mathbf{n}_1} = \lessdot_{\mathbf{n}_3,\mathbf{n}_1}$ is the companion node ordering on node $\mathbf{n}_1$. The intuition behind this ordering is that in order to prove that a state s satisfies U (in node $\mathbf{n}_1$), we also need to establish that all states $s' <:_{\mathbf{n}_1} s$ satisfy U (in node $\mathbf{n}_1$).

It is easy to see that this ordering is well-founded, and thus that leaf $\mathbf{n}_3$ is a successful μ-leaf and the tableau is successful.

The following lemma establishes *pseudo-transitivity* properties of the dependency relations $\lessdot_{\mathbf{n}',\mathbf{n}}$ and $<:_{\mathbf{n}',\mathbf{n}}$. Generally speaking, we would not expect these to be transitive, because e.g., when $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$ holds, s' and s may belong to different sets (namely, the sets of states in their respective proof nodes). However, if we allow the node labels on the relations to align properly, we do have a property that resembles transitivity. The proof of the lemma follows by induction on the definitions. The full proof is included in the appendix.

Lemma 6.19 (Pseudo-transitivity of $\lessdot_{\mathbf{n}',\mathbf{n}}$ and $<:_{\mathbf{n}',\mathbf{n}}$). *Let $\mathbf{n}_1, \mathbf{n}_2$ and $\mathbf{n}_3$ be proof nodes in partial tableau $\mathbb{T}$. The following holds.*

(1) *If* $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_2} s_2$ *and* $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$, *then* $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_1} s_1$.
(2) *If* $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ *and* $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$, *then* $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.
(3) *If* $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ *and* $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_1$, *then* $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.

We now formalize a semantic property, which we call *semantic sufficiency*, enjoyed by the local dependency relation $<_{\mathbf{n}',\mathbf{n}}$ for internal node $\mathbf{n}$. This property asserts that if for every $s \in st(\mathbf{n})$, and every s' such that that $s' <_{\mathbf{n}',\mathbf{n}} s$, s' belongs to the semantics of $\mathbf{n}'$, then this is sufficient to conclude that s belongs to the semantics of $\mathbf{n}$.

Lemma 6.20 (Semantic Sufficiency of $<_{\mathbf{n}',\mathbf{n}}$). *Let* $\mathbf{n}$ *be an internal proof node in partial tableau* $\mathbb{T}$, *and let* $s \in st(\mathbf{n})$ *be such that for all* s' *and* $\mathbf{n}'$ *with* $s' <_{\mathbf{n}',\mathbf{n}} s$, $s' \in ||\,\mathbf{n}'\,||$. *Then* $s \in ||\,\mathbf{n}\,||$.

Proof. Let $\mathbf{n}$ be an internal node in $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$. Since $\mathbf{n}$ is internal, $\rho(\mathbf{n})$ is defined. Now assume $s \in st(\mathbf{n})$; the proof proceeds by a straightforward case analysis on $\rho(\mathbf{n})$. □

7 SOUNDNESS USING SUPPORT STRUCTURES

In this section we prove soundness of the proof system in the previous section by showing that for any successful tableau whose root is labeled by sequent $\mathbf{s} = S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \Phi$, sequent $\mathbf{s}$ must be valid. Our proof relies on establishing that the transitive closure, $<:_{\mathbf{m}}^{+}$, of the companion-node dependency relation (Definition 6.16) used to define the success of μ-leaves is a σ-compatible support structure for the semantic functions associated with fixpoint nodes $\mathbf{m}$. Our reliance on support structures for soundness stands in contrast to Bradfield's and Stirling's soundness proof for essentially this proof system [11, 13], which relies on infinitary logic and the introduction of (infinite) ordinal-unfoldings of fixpoint formulas in particular. Since our ultimate goal in this article is to reason about timed extensions of the modal mu-calculus, we have opted for a different proof strategy that is based more on semantic rather than syntactic reasoning. We also note that our use of support structures is likely to enable the study of other success criteria besides $<:_{\mathbf{m}}$, which is especially interesting for adaptations of this proof system to other settings, such as ones in which formulas are defined equationally or in which less aggressive use is made of formula unfolding than is the case here.

Our proof of soundness proceeds in four steps.

(1) We show (Section 7.1) that tableau rules are *locally sound*, i.e., that when the child nodes of a proof node are valid, then the node itself is also valid.
(2) We wish to be able to reason using tree induction about the meanings of proof nodes and the formulas in those nodes. This reasoning is sometimes impeded because, due to unfolding, many nodes have the same formulas in them. To address this problem, we show how syntactically distinct *node formulas* can be constructed in a semantics-preserving fashion for nodes in a proof tree based on the structure of the proof tree. This material is in Section 7.2.
(3) We then prove that for companion nodes $\mathbf{m}$ in a tableau, the ordering $<:_{\mathbf{m}}^{+}$ is a support structure for the semantic function associated with its node formula. This is done in Section 7.3.
(4) In Section 7.4, we combine the previous three results to obtain soundness for the proof system: if there is a successful tableau whose root sequent $\mathbf{s}$ is such that $dl(\mathbf{s}) = \varepsilon$ then $\mathbf{s}$ is valid.

7.1 Local Soundness

We call a proof system like ours *locally sound* if for every internal node $\mathbf{n}$ in any partial tableau, the validity of all the children of $\mathbf{n}$ implies the validity of $\mathbf{n}$. This may be proven as follows:

Lemma 7.1 (Local Soundness). *Let* $\mathbf{n}$ *be an internal proof node in partial tableau* $\mathbb{T}$. *Then* $\mathbf{n}$ *is valid if all its children are valid.*

Proof. Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, be a partial tableau with internal node $\mathbf{n}$, and assume that for each $\mathbf{n}' \in c(\mathbf{n})$, node $\mathbf{n}'$ is valid. To establish that $\mathbf{n}$ is valid, we must show that $st(\mathbf{n}) \subseteq \|\mathbf{n}\|$. To do so, we fix $s \in st(\mathbf{n})$ and show that $s \in \|\mathbf{n}\|$. In support of this, consider arbitrary $s', \mathbf{n}'$ such that $s' <_{\mathbf{n}',\mathbf{n}} s$ in $\mathbb{T}$. By definition of $<_{\mathbf{n}',\mathbf{n}}$ it follows that $\mathbf{n}' \in c(\mathbf{n})$ and that $s' \in st(\mathbf{n}')$. Moreover, since $\mathbf{n}'$ is valid it follows that $s' \in \|\mathbf{n}'\|$; since this holds for all such s' and $\mathbf{n}'$, Lemma 6.20 ensures that $s \in \|\mathbf{n}\|$. □

7.2 Node Formulas

We now show how to associate a formula $P(\mathbf{n})$ with every node $\mathbf{n}$ in a tableau so that the structure of $P(\mathbf{n})$ is based on the structure of the sub tableau rooted at $\mathbf{n}$ and $P(\mathbf{n})$ is disentangled from the definition list of $\mathbf{n}$. We then show that these formulas are semantically equivalent to the formulas embedded in the nodes' sequents in a precise sense. Since our definition is inductive on the structure of the tableau rooted at $\mathbf{n}$, this facilitates proofs over the semantics of formulas using tree induction.

In the remainder of this section we fix sort Σ, LTS $\mathcal{T} = (\mathcal{S}, \rightarrow)$ over Σ, valuation $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$, and tableau $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$. We also recall the definition of $\mathbf{C}_{\mathbb{T}}$ — the companion nodes of $\mathbb{T}$ — and fix the definitions of the following sets:

$$\mathbb{U} = \bigcup_{\mathbf{n} \in \mathbf{N}} \text{dom}(dl(\mathbf{n})),$$

$$\mathbf{C}_{\mathbf{T}'} = \mathbf{N}' \cap \mathbf{C}_{\mathbb{T}} \text{ for subtree } \mathbf{T}' = (\mathbf{N}', \ldots) \text{ of } \mathbf{T},$$

$$\mathbf{C}_{\mathbf{T}'}(U) = \{\mathbf{n} \in \mathbf{C}_{\mathbf{T}'} \mid fm(\mathbf{n}) = U\} \text{ for subtree } \mathbf{T}' = (\mathbf{N}', \ldots) \text{ of } \mathbf{T} \text{ and } U \in \mathbb{U}.$$

Set $\mathbb{U}$ contains all the definitional constants appearing in $\mathbb{T}$, while $\mathbf{C}_{\mathbf{T}'}$ contains the companion nodes of $\mathbb{T}$ in subtree $\mathbf{T}'$ of $\mathbf{T}$. Set $\mathbf{C}_{\mathbf{T}'}(U)$ consists of the companion nodes of $\mathbb{T}$ in subtree $\mathbf{T}'$ whose formula is $U \in \mathbb{U}$. For subtree $\mathbf{T}_{\mathbf{n}}$ rooted at node $\mathbf{n}$, note that $\mathbf{C}_{\mathbf{T}_\mathbf{n}} \subseteq \mathbf{C}_{\mathbf{T}} = \mathbf{C}_{\mathbb{T}}$. Also, if $\rho(\mathbf{n}) = \text{Un}$ with $c(\mathbf{n}) = \mathbf{n}'$, then $\mathbf{C}_{\mathbf{T}_\mathbf{n}} = \mathbf{C}_{\mathbf{T}_{\mathbf{n}'}} \cup \{\mathbf{n}\}$. Lemma 6.12 and the definition of $\|\mathbf{n}\|$ guarantee, for all $U \in \mathbb{U}$ and $\mathbf{n}, \mathbf{n}' \in \mathbf{C}_{\mathbb{T}}(U)$, that $\|\mathbf{n}\| = \|\mathbf{n}'\|$; we write $\|U\|_{\mathbb{T}}$ for this common value associated with U. We now define $P(-)$ as follows:

Definition 7.2 (Node Formulas). For each companion node $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$ let $Z_{\mathbf{m}}$ be a unique fresh variable, with $\text{Var}_{\mathbb{T}} = \{Z_{\mathbf{m}} \mid \mathbf{m} \in \mathbf{C}_{\mathbb{T}}\}$ the set of all such variables. Then for $\mathbf{n} \in \mathbf{N}$ formula $P(\mathbf{n})$ is defined inductively as follows:

(1) If $\mathbf{n}$ is a free leaf (cf. Definition 6.9(2(a))) then $P(\mathbf{n}) = fm(\mathbf{n})$.
(2) If $\mathbf{n}$ is a $\langle K \rangle$-leaf then $P(\mathbf{n}) = (fm(\mathbf{n}))[dl(\mathbf{n})]$.
(3) If $\mathbf{n}$ is a σ-leaf with companion node $\mathbf{m}$ then $P(\mathbf{n}) = Z_{\mathbf{m}}$.
(4) If $\rho(\mathbf{n}) = \wedge$ and $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ then $P(\mathbf{n}) = P(\mathbf{n}_1) \wedge P(\mathbf{n}_2)$.
(5) If $\rho(\mathbf{n}) = \vee$ and $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ then $P(\mathbf{n}) = P(\mathbf{n}_1) \vee P(\mathbf{n}_2)$.
(6) If $\rho(\mathbf{n}) = [K]$ and $cs(\mathbf{n}) = \mathbf{n}'$ then $P(\mathbf{n}) = [K](P(\mathbf{n}'))$.
(7) If $\rho(\mathbf{n}) = (\langle K \rangle, f)$ and $cs(\mathbf{n}) = \mathbf{n}'$ then $P(\mathbf{n}) = \langle K \rangle(P(\mathbf{n}'))$.
(8) If $\rho(\mathbf{n}) = \sigma Z$ and $cs(\mathbf{n}) = \mathbf{n}'$ then $P(\mathbf{n}) = P(\mathbf{n}')$.
(9) If $\rho(\mathbf{n}) = \text{Thin}$ and $cs(\mathbf{n}) = \mathbf{n}'$ then $P(\mathbf{n}) = P(\mathbf{n}')$.
(10) If $\rho(\mathbf{n}) = \text{Un}$, $\mathbf{n} = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} U$, $\Delta(U) = \sigma Z.\Phi$ and $cs(\mathbf{n}) = \mathbf{n}'$ then $P(\mathbf{n}) = \sigma Z_{\mathbf{n}}.\,(P(\mathbf{n}'))$.

When $\mathbf{n}$ is a free leaf (case 1) $fm(\mathbf{n})$ contains no definitional constants, and thus, $(fm(\mathbf{n}))[dl(\mathbf{n})] = fm(\mathbf{n}) = P(\mathbf{n})$. Also, when $\rho(\mathbf{n}) = \text{Un}$ (case 10), $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ is a companion node in $\mathbb{T}$. Thus, P associates a syntactically distinct formula to each companion node in $\mathbb{T}$.

Intuitively, $P(\mathbf{n})$ can be seen as the formula whose "parse tree" is the sub-tableau of $\mathbb{T}$ rooted at $\mathbf{n}$. The construction works bottom-up from the leaves that are descendants of $\mathbf{n}$, using the proof rule labeling each internal node to recursively construct formulas from those associated with the node's children. Each companion node is converted into a σ-formula, with a freshly generated bound variable that $P(-)$ ensures is assigned to each companion leaf of the companion node. It is easy to see that $P(-)$ contains no instances of any $U \in \mathbb{U}$.

Example 7.3. The node formulas corresponding to the proof tree from Example 6.18 are $P(\mathbf{n}_3) = Z_{\mathbf{n}_1}$, $P(\mathbf{n}_2) = [a]Z_{\mathbf{n}_1}$, and $P(\mathbf{n}_0) = P(\mathbf{n}_1) = \mu Z_{\mathbf{n}_1}.[a]Z_{\mathbf{n}_1}$.

We now turn to establishing a semantic equivalence between $|| \, P(\mathbf{n}) \, ||^{\mathcal{T}}_{\mathcal{V}'}$ for certain $\mathcal{V}'$ and $|| \, \mathbf{n} \, ||$ in $\mathbb{T}$ by first defining the following notion of *valuation consistency with* $\mathbb{T}$.

Definition 7.4 (Valuation Consistency). Let $\mathrm{Var}_{\mathbb{T}}$ be as given in Definition 7.2, and let $\mathcal{V}$ be the valuation in tableau $\mathbb{T}$. Then valuation $\mathcal{V}'$ is *consistent with tableau* $\mathbb{T}$ iff

— for every $U \in \mathbb{U}$ and $\mathbf{m} \in \mathbf{C_T}(U)$, $\mathcal{V}'(Z_{\mathbf{m}}) = || \, U \, ||_{\mathbb{T}}$, and
— for every variable $X \in \mathrm{Var} \setminus \mathrm{Var}_{\mathbb{T}}$, $\mathcal{V}'(X) = \mathcal{V}(X)$.

Intuitively, $\mathcal{V}'$ is consistent with $\mathbb{T}$ iff it assigns the semantics of the associated definitional constant to every fresh variable used in the definition of $P(-)$, and to all other variables it assigns the same value as valuation $\mathcal{V}$ in $\mathbb{T}$. The following result is immediate from the definitions.

Lemma 7.5. *Let $\mathcal{V}'$ be a valuation consistent with $\mathbb{T}$. Then for all $\mathbf{n} \in \mathbf{N}$ such that $fm(\mathbf{n}) = \Phi$ and $dl(\mathbf{n}) = \Delta$, $|| \, \mathbf{n} \, || = || \, \Phi[\Delta] \, ||^{\mathcal{T}}_{\mathcal{V}'}$.*

Proof. Suppose $\mathbf{n} = S \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi$; we must show that $|| \, \mathbf{n} \, || = || \, \Phi[\Delta] \, ||^{\mathcal{T}}_{\mathcal{V}'}$. By definition, $|| \, \mathbf{n} \, || = || \, \Phi \, ||^{\mathcal{T}}_{\mathcal{V}[\Delta]}$. Lemma 6.4 then guarantees that $|| \, \Phi \, ||^{\mathcal{T}}_{\mathcal{V}[\Delta]} = || \, \Phi[\Delta] \, ||^{\mathcal{T}}_{\mathcal{V}}$, and as $\mathcal{V}$ and $\mathcal{V}'$ only disagree on definitional constants in $\mathbb{T}$ that do not appear free in $\Phi[\Delta]$, we have that $|| \, \mathbf{n} \, || = || \, \Phi[\Delta] \, ||^{\mathcal{T}}_{\mathcal{V}'}$. □

We now prove that $|| \, \mathbf{n} \, || = || \, P(\mathbf{n}) \, ||^{\mathcal{T}}_{\mathcal{V}'}$ for proof node $\mathbf{n}$ in $\mathbb{T}$ and $\mathcal{V}'$ consistent with $\mathbb{T}$. This fact establishes that $P(\mathbf{n})$ summarizes all relevant information about the semantics of $\mathbf{n}$, modulo the connection made by $\mathcal{V}'$ between definitional constants in $\mathbf{n}$ and the associated free variables introduced by $P(-)$. The proof is split across two lemmas; we first consider the special case when $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ is a companion node, and then use this result to prove the general case.

Lemma 7.6 (Companion-node Formulas and Semantics). *Let $\mathcal{V}'$ be a valuation that is consistent with tableau $\mathbb{T}$. Then for every $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$, $|| \, P(\mathbf{m}) \, ||^{\mathcal{T}}_{\mathcal{V}'} = || \, \mathbf{m} \, ||$.*

Proof. For any valuation $\mathcal{V}'$ we call a syntactic transformation of Φ to Γ *semantics-preserving* for $\mathcal{V}'$ iff $|| \, \Phi \, ||^{\mathcal{T}}_{\mathcal{V}'} = || \, \Gamma \, ||^{\mathcal{T}}_{\mathcal{V}'}$. Let $\mathcal{V}'$ be consistent with $\mathbb{T}$. We prove the following stronger result:

> for any $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$ with $\mathbf{m} = S \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} U$ and $\Delta(U) = \sigma Z.\Phi$, there is a semantics-preserving transformation of $P(\mathbf{m})$ to $(\sigma Z.\Phi)[\Delta]$ for $\mathcal{V}'$.

The proof of this result proceeds by strong induction on $|\mathbf{C}_{\mathbf{T_m}}|$, the number of companion nodes contained in the subtree $\mathbf{T_m}$ of $\mathbf{T}$ rooted at companion node $\mathbf{m}$. Using this result, one can then establish that $|| \, P(\mathbf{m}) \, ||^{\mathcal{T}}_{\mathcal{V}'} = || \, \sigma Z.\Phi \, ||^{\mathcal{T}}_{\mathcal{V}[\Delta]} = || \, \mathbf{m} \, ||$. □

The next lemma extends the previous one, which focused only on companion nodes, to all nodes.

Lemma 7.7 (Node Formulas and Node Semantics). *Let $\mathcal{V}'$ be a consistent valuation for tableau $\mathbb{T}$. Then for every $\mathbf{n} \in \mathbf{N}$, $|| \, P(\mathbf{n}) \, ||^{\mathcal{T}}_{\mathcal{V}'} = || \, \mathbf{n} \, ||$.*

Proof. The proof proceeds by induction on $\mathbf{T}$, the tree embedded in $\mathbb{T}$. The case where $\rho(\mathbf{n}) = \mathrm{Un}$ follows from Lemma 7.6. □

The final corollary asserts that when the definition list in a proof node is empty, there is no need to make special provision for consistent valuations.

Corollary 7.8. *Let* $\mathbf{n} \in \mathbf{N}$ *be such that* $dl(\mathbf{n}) = \varepsilon$. *Then* $||\,\mathbf{n}\,|| = ||\,P(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}}$.

Proof. Fix $\mathbf{n} \in \mathbf{N}$ such that $dl(\mathbf{n}) = \varepsilon$. Based on Lemma 7.7 we know for any valuation $\mathcal{V}'$ that is consistent with $\mathbb{T}$, that $||\,P(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}'} = ||\,\mathbf{n}\,||$. It may also be seen that no $Z' \in \mathrm{Var}_{\mathbb{T}}$ can be free in $P(\mathbf{n})$, and since $\mathcal{V}'$ is consistent with $\mathbb{T}$ we have that $||\,P(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}'} = ||\,P(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}}$. Consequently, $||\,\mathbf{n}\,|| = ||\,P(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}}$. □

7.3 Support Orderings for Companion Nodes

As the next step in our soundness proof, we establish that for all companion nodes $\mathbf{n}$ in the tableau, $(st(\mathbf{n}), <:^{+}_{\mathbf{n}})$, where $<:^{+}_{\mathbf{n}}$ is the transitive closure of the extended dependency ordering on $\mathbf{n}$, is a support structure for a semantic function derived from $P(\mathbf{n})$. This fact is central in the proof of soundness, as it establishes a key linkage between the tableau-based ordering $<:^{+}_{\mathbf{n}}$ and the semantic notion of support structure.

In order to prove this result about $<:^{+}_{\mathbf{n}}$ we first introduce a derived dependency relation, which we call the *support dependency ordering* (notation $\leq:_{\mathbf{m},\mathbf{n}}$). This ordering is based on the extended dependency ordering $<:_{\mathbf{m},\mathbf{n}}$, but it allows dependencies based on cycling through node $\mathbf{n}$ first, in case $\mathbf{n}$ is a companion node. Specifically, the support dependency ordering captures exactly the dependencies guaranteeing that s is in the semantics of $\mathbf{n}$ if for every s', $\mathbf{m}$ with $s' \leq:_{\mathbf{m},\mathbf{n}} s$, state s' is in the semantics of $\mathbf{m}$. If $\mathbf{n}$ is a node in which the unfolding rule has been applied, to show that s is in the semantics of $\mathbf{n}$, we may require to first show that s' is in the semantics of $\mathbf{n}$. This is not captured by the relation $<:_{\mathbf{m},\mathbf{n}}$, which does not take the dependencies within node $\mathbf{n}$ into account.

Definition 7.9 (Support Dependency Ordering). Let $\mathbf{m}, \mathbf{n} \in \mathbf{N}$ be proof nodes in $\mathbb{T}$. The *support dependency ordering*, $\leq:_{\mathbf{m},\mathbf{n}}$ is defined as follows:

$$\leq:_{\mathbf{m},\mathbf{n}} = \begin{cases} <:_{\mathbf{m},\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}} & \text{if } \rho(\mathbf{n}) = \mathrm{Un} \\ <:_{\mathbf{m},\mathbf{n}} & \text{otherwise.} \end{cases}$$

Example 7.10. Reconsider the proof system of our running example. In Example 6.18, we commented on the orderings. The support dependency ordering $\leq:_{\mathbf{n}_3,\mathbf{n}_1}$ for this example is as follows: $s \leq:_{\mathbf{n}_3,\mathbf{n}_1} \omega$, for all $s \in \mathbb{N}$, and for all $s_1 \in \mathbb{N}$, and $s_2 < s_1$ (where $<$ is the ordering on natural numbers), $s_2 \leq:_{\mathbf{n}_3,\mathbf{n}_1} s_1$. This captures that, in order to prove that ω is in the semantics of node $\mathbf{n}_1$, all natural numbers need to be in the semantics of node $\mathbf{n}_3$. Likewise, for state 2 to be in the semantics of $\mathbf{n}_1$, states 0 and 1 need to be in the semantics of $\mathbf{n}_3$. Note that in order to establish that $0 \leq:_{\mathbf{n}_3,\mathbf{n}_1} 2$, we use that $0 <:_{\mathbf{n}_3,\mathbf{n}_1} 1$, and $1 <:^{*}_{\mathbf{n}_1} 2$. Also, observe that $\leq:_{\mathbf{n}_3,\mathbf{n}_0}$ is the same as $\leq:_{\mathbf{n}_3,\mathbf{n}_1}$.

We now remark on some properties of $\leq:_{\mathbf{m},\mathbf{n}}$ that will be used below. We first note that $\leq:_{\mathbf{m},\mathbf{n}}$ extends $<:_{\mathbf{m},\mathbf{n}}$ (as well as $<_{\mathbf{m},\mathbf{n}}$ and $\lessdot_{\mathbf{m},\mathbf{n}}$, since $<:_{\mathbf{m},\mathbf{n}}$ extends both of these relations): for all s, s', if $s' <:_{\mathbf{m},\mathbf{n}} s$ then $s' \leq:_{\mathbf{m},\mathbf{n}} s$. Also, if $\mathbf{n}$ is a companion node (i.e., $\rho(\mathbf{n}) = \mathrm{Un}$) then the transitivity of $<:^{*}_{\mathbf{n}}$ guarantees that $\leq:_{\mathbf{m},\mathbf{n}} = (\leq:_{\mathbf{m},\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}})$, as in this case

$$\leq:_{\mathbf{m},\mathbf{n}} = (<:_{\mathbf{m},\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}}) = (<:_{\mathbf{m},\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}}) = (\leq:_{\mathbf{m},\mathbf{n}}\ ;\ <:^{*}_{\mathbf{n}}).$$

From the definition of $<:_{\mathbf{m},\mathbf{n}}$ (cf. Definition 6.16) we have that if $\mathbf{m} = \mathbf{n}$ then $<:_{\mathbf{m},\mathbf{n}} = Id_{st(\mathbf{n})}$ is the identity relation over $st(\mathbf{n})$. From this fact we can make the following observations. First, if $\mathbf{m} = \mathbf{n}$ and $\mathbf{n}$ is not a companion node, then $\leq:_{\mathbf{m},\mathbf{n}} = <:_{\mathbf{m},\mathbf{n}}$ is the identity relation over $st(\mathbf{n})$. Second, if $\mathbf{m} = \mathbf{n}$ and $\mathbf{n}$ is a companion node then $\leq:_{\mathbf{m},\mathbf{n}}$ is $<:^{*}_{\mathbf{n}}$, the reflexive and transitive closure of the companion node ordering for $\mathbf{n}$.

Relation $\leq:_{\mathbf{m},\mathbf{n}}$ also enjoys a pseudo-transitivity property.

Lemma 7.11 (Pseudo-transitivity of $\leq:_{\mathbf{m},\mathbf{n}}$). *Let* $\mathbf{n}_1, \mathbf{n}_2$, *and* $\mathbf{n}_3$ *be proof nodes in partial tableau* $\mathbb{T}$, *and assume* s_1, s_2, *and* s_3 *are such that* $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ *and* $s_2 \leq:_{\mathbf{n}_2,\mathbf{n}_1} s_1$. *Then* $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.

In the remainder of this section we wish to establish that for any companion node $\mathbf{n}$ in successful tableau $\mathbb{T}$, $<:^+_{\mathbf{n}}$ is a support structure for a function derived from $P(\mathbf{n})$. In order to define this function, we must deal with the free variables embedded in $P(\mathbf{n})$. In particular, if $\mathbf{m}$ is a companion node that is a strict ancestor of $\mathbf{n}$ then variable $Z_{\mathbf{m}}$ may appear free in $P(\mathbf{n})$; this would be the case if any of the companion leaves of $\mathbf{m}$ are also descendants of $\mathbf{n}$. To accommodate these free variables in $P(\mathbf{n})$ we will define a modification of valuation $\mathcal{V}$ that assigns sets of states to these variables based on the $<:_{\mathbf{m}',\mathbf{n}}$ relation, where $\mathbf{m}'$ is a companion leaf of $\mathbf{m}$ that is also a descendant of $\mathbf{n}$.

Definition 7.12 (Influence Extensions of Valuations). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a tableau, with $\mathbf{m}_1 \cdots \mathbf{m}_k$ an ordering on the companion nodes $\mathbf{C}_{\mathbb{T}}$ of $\mathbb{T}$. Also let $\mathbf{n}$ be a node in $\mathbb{T}$, with $S = st(\mathbf{n})$ the states in $\mathbf{n}$. We define the following:

(1) $\mathbf{CL}_{\mathbf{m}_i,\mathbf{n}} = \mathbf{CL}_{\mathbb{T}}(\mathbf{m}_i) \cap D(\mathbf{n})$ is the set of companion leaves of $\mathbf{m}_i$ that are also descendants of $\mathbf{n}$.
(2) The set of states in companion leaves of $\mathbf{m}_i$ that influence state s in $\mathbf{n}$ is given as follows:

$$S_{\mathbf{n},s,\mathbf{m}_i} = \bigcup_{\mathbf{m}' \in \mathbf{CL}_{\mathbf{m}_i,\mathbf{n}}} (\leq:_{\mathbf{m}',\mathbf{n}})^{-1}(s).$$

We also define

$$S_{\mathbf{n},\mathbf{m}_i} = \bigcup_{s \in S} S_{\mathbf{n},s,\mathbf{m}_i}$$

to be the set of states in companion leaves of $\mathbf{m}_i$ that influence $\mathbf{n}$.
(3) The *influence extension* of $\mathcal{V}$ for state s in node $\mathbf{n}$ is defined as

$$\mathcal{V}_{\mathbf{n},s} = \mathcal{V}[Z_{\mathbf{m}_1} \cdots Z_{\mathbf{m}_k} = S_{\mathbf{n},s,\mathbf{m}_1} \cdots S_{\mathbf{n},s,\mathbf{m}_k}].$$

Similarly

$$\mathcal{V}_{\mathbf{n}} = \mathcal{V}[Z_{\mathbf{m}_1} \cdots Z_{\mathbf{m}_k} = S_{\mathbf{n},\mathbf{m}_1} \cdots S_{\mathbf{n},\mathbf{m}_k}]$$

is the influence extension of $\mathcal{V}$ for node $\mathbf{n}$.

Intuitively, $S_{\mathbf{n},s,\mathbf{m}_i}$ contains all the states in the companion leaves of $\mathbf{m}_i$ at or below node $\mathbf{n}$ that influence the determination that state s belongs in node $\mathbf{n}$. Note these definitions also set $\mathcal{V}_{\mathbf{n},s}(Z_{\mathbf{m}_i}) = \emptyset$ in case node $\mathbf{m}_i$ has no companion leaves that are descendants of $\mathbf{n}$; when this happens $Z_{\mathbf{m}_i}$ cannot appear free in $P(\mathbf{n})$. Also note that $Z_{\mathbf{m}_i}$ does not appear free in $P(\mathbf{n})$ if $\mathbf{m}_i$ is a descendant of $\mathbf{n}$, as $P(\mathbf{m}_i) = \sigma Z_{\mathbf{m}_i}.\Phi'$ for some Φ' is a subformula of $P(\mathbf{n})$ and contains all occurrences of $Z_{\mathbf{m}_i}$ in $P(\mathbf{n})$. In both cases the value assigned to $Z_{\mathbf{m}_i}$ by $\mathcal{V}_{\mathbf{n},s}$ does not affect the semantics of $P(\mathbf{n})$.

Example 7.13. Consider the proof tree from Example 6.18. We described the support dependency ordering in Example 7.10. The set of states in the companion leaves of $\mathbf{n}_1$ in the tree (rooted at $\mathbf{n}_0$) that influence state ω are $S_{\mathbf{n}_0,\omega,\mathbf{n}_1} = (\leq:_{\mathbf{n}_3,\mathbf{n}_0})^{-1}(\omega) = \mathbb{N}$. The set $S_{\mathbf{n}_1,\omega,\mathbf{n}_1}$ is the same. The influence extensions for state ω are $\mathcal{V}_{\mathbf{n}_0,\omega} = \mathcal{V}_{\mathbf{n}_1,\omega} = \mathcal{V}[Z_{\mathbf{n}_1} = S_{\mathbf{n}_0,\omega,\mathbf{n}_1}] = \mathcal{V}[Z_{\mathbf{n}_1} = \mathbb{N}]$.

We now state a technical but useful lemma about dependency extensions.

Lemma 7.14 (Monotonicity of Extensions). *Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a tableau with nodes $\mathbf{n}$ and $\mathbf{n}'$ and states s and s' such that $s' <_{\mathbf{n}',\mathbf{n}} s$. Then:*

(1) for all $Z \in \mathrm{Var}_{\mathbb{T}}$, $\mathcal{V}_{\mathbf{n}',s'}(Z) \subseteq \mathcal{V}_{\mathbf{n},s}(Z)$, and
(2) for all $Z \in \mathrm{Var} \setminus \mathrm{Var}_{\mathbb{T}}$, $\mathcal{V}_{\mathbf{n}',s'}(Z) = \mathcal{V}_{\mathbf{n},s}(Z)$.

Proof. Follows from the definition of $\mathcal{V}_{\mathbf{n},s}$ and the fact that the definition of $<_{\mathbf{n}',\mathbf{n}}$ ensures that $\mathbf{n}' \in c(\mathbf{n})$ and thus $D(\mathbf{n}') \subseteq D(\mathbf{n})$. Consequently $\mathbf{CL}_{\mathbf{m}_i,\mathbf{n}'} \subseteq \mathbf{CL}_{\mathbf{m}_i,\mathbf{n}}$ for all $\mathbf{m}_i \in \mathbf{C}_{\mathbb{T}}$, and $s' <_{\mathbf{n}',\mathbf{n}} s$ guarantees that $S_{\mathbf{n}',s',\mathbf{m}_i} \subseteq S_{\mathbf{n},s,\mathbf{m}_i}$. □

The next corollary is an immediate consequence of this lemma and the fact that every occurrence of any $Z_{\mathbf{m}} \in \mathrm{Var}_{\mathbb{T}}$ in $P(\mathbf{n}')$ must be positive.

Corollary 7.15. *Let* $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ *be a tableau, with* $\mathbf{n}$ *and* $\mathbf{n}'$ *and states* s *and* s' *such that* $s' <_{\mathbf{n}',\mathbf{n}} s$. *Then* $||\, P(\mathbf{n})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{n}',s'}} \subseteq ||\, P(\mathbf{n})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{n}',s}}$.

We now state and prove the main lemma of this section, which is that for any companion node $\mathbf{n}$ in a successful tableau, $<:^{+}_{\mathbf{n}}$ is a support structure for a semantic function derived from $P(\mathbf{n})$.

Lemma 7.16 ($<:^{+}_{\mathbf{n}}$ is a Support Structure). *Let* $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ *be a successful tableau, with* $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ *a companion node of* $\mathbb{T}$ *and* $\mathbf{n}'$ *the child of* $\mathbf{n}$ *in* $\mathbf{T}$. *Also let* $S = st(\mathbf{n})$. *Then* $(S, <:^{+}_{\mathbf{n}})$ *is a support structure for* $||\, Z_{\mathbf{n}}.P(\mathbf{n}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{n}}}$.

Proof. Fix successful tableau $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, and let $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ be a companion node of $\mathbb{T}$ with $S = st(\mathbf{n})$. We prove the following statements for all $\mathbf{m} \in D(\mathbf{n})$ and $s \in S$; this implies the lemma.

S1. For all x such that $x \leq:_{\mathbf{m},\mathbf{n}} s$, $x \in ||\, P(\mathbf{m})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$.
S2. If $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$, $\mathbf{m}' = cs(\mathbf{m})$ and x satisfies $x \leq:_{\mathbf{m},\mathbf{n}} s$ then $(S_x, <:_{\mathbf{m},x})$ is a support structure for $||\, Z_{\mathbf{m}}.P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$, where $S_x = (<:^{*}_{\mathbf{m}})^{-1}(x)$ and $<:_{\mathbf{m},x} = (<:^{+}_{\mathbf{m}})\lfloor S_x$.

The proof proceeds by tree induction on $\mathbf{T}_{\mathbf{n}}$, the subtree rooted at $\mathbf{n}$ in $\mathbf{T}$. To this end, fix $\mathbf{m} \in D(\mathbf{n})$; the induction hypothesis states that for all $\mathbf{m}' \in D{\downarrow}(\mathbf{m})$ and $s \in S$, S1 and S2 hold. We must show that for any $s \in S$, S1 and S2 are true of $\mathbf{m}$. So fix $s \in S$. We now do a case split on whether $\rho(\mathbf{m}) = \mathrm{Un}$. If $\rho(\mathbf{m}) \neq \mathrm{Un}$ then $\mathbf{m} \notin \mathbf{C}_{\mathbb{T}}$, and S2 vacuously holds for all $s \in S$; all that needs to be proved is S1 for $\mathbf{m}$ and s. We now do a case analysis on all the forms of $\rho(\mathbf{m})$ when $\rho(\mathbf{m}) \neq \mathrm{Un}$. All such cases are similar; we only consider $\rho(\mathbf{n}) = (\langle K\rangle, f)$ in detail. In this case $\mathbf{m} = X \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \langle K\rangle\Phi$ for some X and Φ, $cs(\mathbf{m}) = \mathbf{m}'$, and $\mathbf{m}' = f(X) \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi$; recall that $x \xrightarrow{K} f(x)$ for all $x \in X$. The induction hypothesis ensures that for all $t \in S$, S1 holds for $\mathbf{m}'$; we must show that S1 holds for $\mathbf{m}$ and s. To this end, let $x \in X$ be such that $x \leq:_{\mathbf{m},\mathbf{n}} s$; we must show that $x \in ||\, P(\mathbf{m})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. Note that $f(x) <_{\mathbf{m}',\mathbf{m}} x$; the pseudo-transitivity of $\leq:_{\mathbf{m},\mathbf{n}}$ then guarantees that $f(x) \leq:_{\mathbf{m}',\mathbf{n}} s$, and the induction hypothesis ensures that $f(x) \in ||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}',f(x)}}$. Corollary 7.15 implies that $f(x) \in ||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},f(x)}}$, and the semantics of $\langle K\rangle$ establish that $x \in ||\, \langle K\rangle P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}} = ||\, P(\mathbf{m})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$; thus S1 holds for $\mathbf{m}$ and s.

Now suppose $\rho(\mathbf{m}) = \mathrm{Un}$. In this case $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$, meaning $\mathbf{m} = X \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} U$, where $U \in \mathrm{dom}(\Delta)$, $\Delta(U) = \sigma Z.\Phi$, $cs(\mathbf{m}) = \mathbf{m}'$ and $\mathbf{m}' = X \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi[Z := U]$. We must show S1 and S2 for $\mathbf{m}$ and s. We consider S2 first. Let $x \leq:_{\mathbf{m},\mathbf{n}} s$, and define $f_{\mathbf{m},x} = ||\, Z_{\mathbf{m}}.P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. We must show that $(S_x, <:_{\mathbf{m},x})$ is a support structure for $f_{\mathbf{m},x}$. Following Definition 4.1, it suffices to prove that for every $x' \in S_x$, $x' \in f_{\mathbf{m},x}((<:_{\mathbf{m},x})^{-1}(x'))$. So fix $x' \in S_x$. By definition of S_x this means that $x' <:^{*}_{\mathbf{m}} x$. Since $x' <_{\mathbf{m}',\mathbf{m}} x'$, it follows that $x' \leq:_{\mathbf{m}',\mathbf{m}} x'$ and, due to the pseudo-transitivity Lemma 7.11, that $x' \leq:_{\mathbf{m}',\mathbf{n}} s$. From the induction hypothesis, we know that S1 holds for $\mathbf{m}'$ and x', meaning $x' \in ||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}',x'}}$. To complete this part of the proof it suffices to establish that $||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}',x'}} \subseteq f_{\mathbf{m},x}((<:_{\mathbf{m},x})^{-1}(x'))$. We begin by noting that since $x' <_{\mathbf{m}',\mathbf{m}} x'$ and all occurrences of any $Z \in \mathrm{Var}_{\mathbb{T}}$ in $P(\mathbf{m}')$ are positive, Lemma 7.14 ensures that $||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}',x'}} \subseteq ||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x'}}$. It, therefore, suffices to show that $||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x'}} \subseteq f_{\mathbf{m},x}((<:_{\mathbf{m},x})^{-1}(x'))$. From the definition of $f_{\mathbf{m},x}$,

$$f_{\mathbf{m},x}((<:_{\mathbf{m},x})^{-1}(x')) = ||\, P(\mathbf{m}')\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}[Z_{\mathbf{m}}:=(<:_{\mathbf{m},x})^{-1}(x')]}.$$

Because every $Z \in \mathrm{Var}_{\mathbb{T}}$ appearing in $P(\mathbf{m}')$ appears only positively, the fact that $\|\, P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x'}} \subseteq f_{\mathbf{m},x}((<:_{\mathbf{m},x})^{-1}(x'))$ follows from the following two observations.

(1) For all $Z \in \mathrm{Var} \setminus \mathrm{Var}_{\mathbb{T}}$, $\mathcal{V}_{\mathbf{m},x'}(Z) = \left(\mathcal{V}_{\mathbf{m},x}[Z_{\mathbf{m}} := (<:_{\mathbf{m},x})^{-1}(x')]\right)(Z)$.
(2) For all $Z \in \mathrm{Var}_{\mathbb{T}}$, $\mathcal{V}_{\mathbf{m},x'}(Z) \subseteq \left(\mathcal{V}_{\mathbf{m},x}[Z_{\mathbf{m}} := (<:_{\mathbf{m},x})^{-1}(x')]\right)(Z)$.

To finish the proof we now need to show that statement S1 holds for companion node $\mathbf{m}$ and $s \in S$. So fix x such that $x \leq:_{\mathbf{m},\mathbf{n}} s$; we must show that $x \in \|\, P(\mathbf{m}) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. We know from the definition of P that $P(\mathbf{m}) = \sigma Z_{\mathbf{m}}.P(\mathbf{m}')$. If $\sigma = \nu$ then as statement S2 holds we know that $(S_x, <:_{\mathbf{m},x})$ is a support structure for $\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. Therefore, S_x is supported for $\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$, and Corollary 4.7 ensures that $S_x \subseteq \nu(\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}})$. As $x \in S_x$ and $\nu(\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}) = \|\, P(\mathbf{m}) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$, the result follows. The case where $\sigma = \mu$ follows a similar line of reasoning, but uses the fact that $<:_{\mathbf{m},x}$ is well-founded to note that S_x is well-supported for $\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. Corollary 4.5 then implies the desired result. □

Corollary 7.17 (Support Structures for Top-level Companion Nodes). *Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a successful tableau, with $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ a top-level companion of $\mathbb{T}$ and $\mathbf{n}'$ the child of $\mathbf{n}$ in $\mathbf{T}$. Also let $S = st(\mathbf{n})$. Then $(S, <:^{+}_{\mathbf{n}})$ is a support structure for $\|\, Z_{\mathbf{n}}.P(\mathbf{n}') \,\|^{\mathcal{T}}_{\mathcal{V}}$.*

Proof. Follows from the fact that since $\mathbf{n}$ is top-level, the only variable in $\mathrm{Var}_{\mathbb{T}}$ that can appear free in $P(\mathbf{n}')$ is $Z_{\mathbf{n}}$. Lemma 7.15 thus guarantees that for any S', $\|\, Z_{\mathbf{n}}.P(\mathbf{n}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{n}}}(S') = \|\, Z_{\mathbf{n}}.P(\mathbf{n}') \,\|^{\mathcal{T}}_{\mathcal{V}}(S')$. Lemma 7.16 then establishes the corollary. □

Example 7.18. Recall node ordering $<:_{\mathbf{n}_1} = \{(s, \omega) \mid s \in \mathbb{N}\} \cup \{(s, s+1) \mid s \in \mathbb{N}\}$ from Example 6.18. It follows that $<:^{+}_{\mathbf{n}_1} = \{(s, \omega) \mid s \in \mathbb{N}\} \cup \{(s, s') \mid s, s' \in \mathbb{N} \text{ and } s < s'\}$. The structure $(\mathbb{N} \cup \{\omega\}, <:^{+}_{\mathbf{n}_1})$ is a well-founded support structure for the function $\|\, Z_{\mathbf{n}_1}.[a]Z_{\mathbf{n}_1} \,\|^{\mathcal{T}}_{\mathcal{V}}$ in our running example.

7.4 Soundness

We now prove that our proof system is sound by establishing that the root sequent of every successful tableau is valid. This proof relies on first constructing the *prefix* of a successful tableaux, in which the top-level companion nodes in this tableaux are turned into leaves. Formally, this prefix is obtained as follows:

Definition 7.19 (Tree-prefix Generation). Let $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ be a finite ordered tree, and let $\mathbf{L} \subseteq \mathbf{N}$. Then $\mathbf{T}\lceil\mathbf{L}$, the *tree prefix of* $\mathbf{T}$ *generated by* $\mathbf{L}$, is the tree $(\mathbf{N}', \mathbf{r}, p', cs')$ given as follows:

— $\mathbf{N}' = \mathbf{N} \setminus (\bigcup_{\mathbf{l} \in \mathbf{L}} D{\downarrow}(\mathbf{l}))$.
— Let $\mathbf{n}' \in \mathbf{N}'$. Then $p'(\mathbf{n}') = p(\mathbf{n}')$.
— Let $\mathbf{n}' \in \mathbf{N}'$. Then $cs'(\mathbf{n}') = \varepsilon$ if $D{\downarrow}(\mathbf{n}) \cap \mathbf{N}' = \emptyset$, and $cs(\mathbf{n}')$, otherwise.

The nodes of $\mathbf{T}\lceil\mathbf{L}$ are the nodes of $\mathbf{T}$ with the strict descendants of nodes in $\mathbf{L}$ removed. While $\mathbf{L}$ can be thought of as specifying nodes in $\mathbf{T}$ that should be converted into leaves in $\mathbf{T}\lceil\mathbf{L}$, this intuition is only partially accurate, since it is not the case that every $\mathbf{l} \in \mathbf{L}$ is a node in $\mathbf{T}\lceil\mathbf{L}$. In particular, if $\mathbf{l}$ has a strict ancestor in $\mathbf{L}$ this would cause the removal of $\mathbf{l}$ from $\mathbf{T}\lceil\mathbf{L}$. However, if $\mathbf{l} \in \mathbf{L}$ has no strict ancestors in $\mathbf{L}$ then it is indeed a leaf in $\mathbf{T}\lceil\mathbf{L}$.

Theorem 7.20 (Soundness of Mu-calculus Proof System). *Fix LTS $(\mathcal{S}, \rightarrow)$ of sort Σ and valuation $\mathcal{V}$, and let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a successful tableau for sequent $\mathbf{s} \in \mathbf{S}^{\mathcal{T}}_{\mathrm{Var}}$, where $dl(\mathbf{s}) = \varepsilon$. Then $\mathbf{s}$ is valid.*

Proof. Let $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ be the tree component of $\mathbb{T}$, and define $\mathbf{L} \subseteq \mathbf{N}$ as follows:

$$\mathbf{L} = \{\mathbf{n} \in \mathbf{N} \mid dl(\mathbf{n}) = \varepsilon \wedge \rho(\mathbf{n}) = \sigma Z\}.$$

Now consider the tree prefix $\mathbf{T}\lceil\mathbf{L}$ of $\mathbf{T}$. It can be seen that $\mathbf{T}\lceil\mathbf{L} = (\mathbf{N}', \mathbf{r}, p', cs')$ is such that $\mathbf{N}'$ contains precisely the nodes of $\mathbf{T}$ for which $dl(\mathbf{n}) = \varepsilon$. Moreover, each leaf $\mathbf{n}$ of $\mathbf{T}\lceil\mathbf{L}$ is either a leaf of $\mathbf{T}$ or has the property that $\mathbf{n} = S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi$ for some S and Φ and that the child $\mathbf{n}'$ of $\mathbf{n}$ in $\mathbf{T}$ is such that $\mathbf{n}' = S \vdash_{(U=\sigma Z.\Phi)}^{\mathcal{T},\mathcal{V}} U$ is a top-level companion node in $\mathbb{T}$. We will show that each leaf $\mathbf{n}$ of $\mathbf{T}\lceil\mathbf{L}$ is valid; this fact, and Lemma 7.1, can be used as the basis for a simple inductive argument on $\mathbf{T}\lceil\mathbf{L}$ to establish that every node in $\mathbf{T}\lceil\mathbf{L}$ is valid, including root node $\mathbf{r}$, whose sequent label is $\mathbf{s}$.

So fix leaf $\mathbf{n}$ in $\mathbf{T}\lceil\mathbf{L}$. There are two cases to consider. In the first, $\mathbf{n}$ is also a leaf in $\mathbf{T}$. In this case, since $\mathbb{T}$ is successful, $\mathbf{n}$ is successful, and, therefore, valid. In the second case, $\mathbf{n} = S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi$ and has a single child $\mathbf{n}' = S \vdash_{(U=\sigma Z.\Phi)}^{\mathcal{T},\mathcal{V}} U$ that is a top-level companion node of $\mathbb{T}$. Let $\mathbf{n}''$ be the child of $\mathbf{n}'$. Corollary 7.17 guarantees that $(S, <:_{\mathbf{n}'}^{+})$ is a support structure for $|| Z_{\mathbf{n}'}.P(\mathbf{n}'') ||_{\mathcal{V}}^{\mathcal{T}}$. We will now show that $S \subseteq || P(\mathbf{n}') ||_{\mathcal{V}}^{\mathcal{T}}$. There are two sub-cases to consider. In the first, $\sigma = \nu$. It follows from the definitions that S is supported for $|| Z_{\mathbf{n}'}.P(\mathbf{n}'') ||_{\mathcal{V}}^{\mathcal{T}}$ and thus by Corollary 4.7, $S \subseteq \nu(|| Z_{\mathbf{n}'}.P(\mathbf{n}'') ||_{\mathcal{V}}^{\mathcal{T}}) = || P(\mathbf{n}') ||_{\mathcal{V}}^{\mathcal{T}}$. In the second, $\sigma = \mu$. Since $\mathbb{T}$ is successful $<:_{\mathbf{n}'}$ is well-founded, meaning that $<:_{\mathbf{n}'}^{+}$ is also well-founded. Thus, S is well-supported for $|| Z_{\mathbf{n}'}.P(\mathbf{n}'') ||_{\mathcal{V}}^{\mathcal{T}}$ and thus by Corollary 4.5, $S \subseteq \mu(|| Z_{\mathbf{n}'}.P(\mathbf{n}'') ||_{\mathcal{V}}^{\mathcal{T}}) = || P(\mathbf{n}') ||_{\mathcal{V}}^{\mathcal{T}}$. Since $P(\mathbf{n}) = P(\mathbf{n}')$, we know $S \subseteq || P(\mathbf{n}) ||_{\mathcal{V}}^{\mathcal{T}}$. Furthermore, note that since $\mathbf{n}'$ is a top-level companion node, $P(\mathbf{n})$ can contain no free occurrences of any $Z' \in \mathrm{Var}_{\mathbb{T}}$, meaning that for any $\mathcal{V}'$ consistent with $\mathbb{T}$, $|| P(\mathbf{n}) ||_{\mathcal{V}}^{\mathcal{T}} = || P(\mathbf{n}) ||_{\mathcal{V}'}^{\mathcal{T}}$. Consequently, Lemma 7.6 implies that $|| P(\mathbf{n}) ||_{\mathcal{V}}^{\mathcal{T}} = || \mathbf{n} ||$, and thus $S \subseteq || \mathbf{n} ||$, whence $\mathbf{n}$ is valid. □

8 COMPLETENESS

This section now establishes the completeness of our proof system. Call a tableau $(\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$, where $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, *successful for* sequent $S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$ iff it is successful and $\mathbf{r} = S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \Phi$. We show that for any $\mathcal{T}$, $\mathcal{V}$, S and Φ, if $S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \Phi$ is valid then there is a successful tableau for $S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \Phi$.

The completeness results in this section rely heavily on tableau manipulations; in particular, several proofs define constructions for merging multiple successful tableaux into a single successful tableau. These constructions in turn rely on variations of well-founded induction over support structures for the semantic functions used to give meaning to fixpoint formulas, and become subtle in the setting of mutually recursive fixpoints. To clarify and simplify these arguments, the first subsection below introduces relevant notions from general fixpoint theory in the setting of mutual recursion. These results are then used later in this section to define the tableau constructions we need to establish completeness.

8.1 Mutual Recursion and Fixpoints

Mu-calculus formulas of form $\sigma Z.(\cdots \sigma' Z'.(\cdots Z \cdots) \cdots)$ are said to be *mutually recursive*, because the semantics of the outer fixpoint formula, $\sigma Z. \cdots$, depends on the semantics of the inner fixpoint formula, $\sigma' Z'. \cdots$, which in turn (because Z is free in its body) depends on the semantics of the outer fixpoint. If $\sigma \neq \sigma'$ then these mutually recursive fixpoints are also said to be *alternating*. Mutually recursive fixpoints present challenges when reasoning about completeness. In this section we develop a theory of mutually recursive fixpoints in the general setting of recursive functions over complete lattices to support this reasoning. In particular, we show how to define mutually recursive fixpoints in terms of binary functions, and we define a property of binary relations, which we call *quotient well-foundedness*, that can be applied to support structures for mutually recursive fixpoints in order to support a form of well-founded induction.

Mutual recursion. Let S be a set, with $(2^S, \subseteq, \bigcup, \bigcap)$ the subset lattice over S (cf. Definition 3.17). To formalize mutually recursive fixpoints we use *monotonic binary* functions over 2^S.

Definition 8.1 (Monotonic Binary Functions). Binary function $f \in 2^S \times 2^S \rightarrow 2^S$ is *monotonic* iff for all $X_1, X_2, Y_1, Y_2 \in 2^S$, if $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$ then $f(X_1, Y_1) \subseteq f(X_2, Y_2)$.

Binary functions are monotonic when they are monotonic in each argument individually. We now define the following operations on binary functions to help formalize mutually recursive fixpoints.

Definition 8.2 (Binary-function Operations). Let $f \in 2^S \times 2^S \rightarrow 2^S$ be monotonic.

(1) Let $X, Y \subseteq S$. Then functions $f_{(X,\cdot)}, f_{(\cdot,Y)} \in 2^S \rightarrow 2^S$ are defined by

$$f_{(X,\cdot)}(Y) = f_{(\cdot,Y)}(X) = f(X, Y).$$

Since $f_{(X,\cdot)}$ and $f_{(\cdot,Y)}$ are monotonic when f is, we may further define $f_{(\cdot,\sigma)}, f_{(\sigma,\cdot)} \in 2^S \rightarrow 2^S$, where $\sigma \in \{\mu, \nu\}$, as follows:

$$f_{(\cdot,\sigma)}(X) = \sigma f_{(X,\cdot)},$$
$$f_{(\sigma,\cdot)}(Y) = \sigma f_{(\cdot,Y)}.$$

(2) Suppose further that $g \in 2^S \times 2^S \rightarrow 2^S$ is binary and monotonic over 2^S and that $\sigma \in \{\mu, \nu\}$. Then the function $(f[\sigma]g) \in 2^S \rightarrow 2^S$ is defined as follows:

$$(f[\sigma]g)(X) = f(X, g_{(\cdot,\sigma)}(X)).$$

To understand the above definitions, first note that if f is binary and monotonic then $f_{(X,\cdot)} \in 2^S \rightarrow 2^S$ is the unary function obtained by holding the first argument of f fixed at X, leaving only the second argument to vary. Similarly, $f_{(\cdot,Y)} \in 2^S \rightarrow 2^S$ is the unary function obtained by holding the second argument of f fixed at Y. The monotonicity of f guarantees the monotonicity of $f_{(X,\cdot)}$ and $f_{(\cdot,Y)}$, and thus fixpoints $\sigma f_{(X,\cdot)}$ and $\sigma f_{(\cdot,Y)}$ are well defined for all X and Y and $\sigma \in \{\mu, \nu\}$. This fact ensures that unary functions $f_{(\cdot,\sigma)}$ and $f_{(\sigma,\cdot)}$ are well-defined; in the first case, given argument X, $f_{(\cdot,\sigma)}(X)$ returns the result of computing the σ-fixpoint of f when the first argument of f is held at X. The second case is similar. It is straightforward to establish that $f_{(\cdot,\sigma)}$ and $f_{(\sigma,\cdot)}$ are also monotonic for all $\sigma \in \{\mu, \nu\}$. Finally, for each $\sigma \in \{\mu, \nu\}$ the operation $[\sigma]$ defines a composition operation that converts binary monotonic functions f and g into a unary monotonic function with the following behavior. Given input $X \subseteq S$ the composition function applies f to X and the result of computing the σ-fixpoint of g with its first argument held at X. To understand the importance of this function, consider the following notional pair of mutually recursive equations, where f and g are monotonic binary functions.

$$X \overset{\sigma}{=} f(X, Y),$$
$$Y \overset{\sigma'}{=} g(X, Y).$$

Here $\sigma, \sigma' \in \{\mu, \nu\}$; the intention of these equations is to define X and Y as the mutually recursive σ and σ' fixpoints of f and g, with the first equation dominating the second one. In the usual definitions of such equation systems, X is defined to be $\sigma(f[\sigma']g)$, i.e., the σ-fixpoint of the function $f[\sigma']g$, while Y is taken to be $g_{(\cdot,\sigma')}(X)$; see, e.g., [50].

The next lemma highlights the role that the $f[\sigma]g$ construct plays in the semantics of the mu-calculus formulas that involve nested fixpoints. The statement relies on the notion of a *maximal fixpoint subformula*. Briefly, if Φ is a formula then $\sigma Z.\Gamma$ is a maximal fixpoint subformula of Φ iff it is a subformula of Φ and is not a proper subformula of any other fixpoint subformula of Φ.

Lemma 8.3 (Nested Fixpoint Semantics). *Let $\sigma Z.\Phi \in \mathbb{F}^{\Sigma}_{\text{Var}}$ be a formula, let $\sigma' Z'.\Gamma$ be a maximal fixpoint subformula of Φ, and let Φ' and $W \in$ Var be such that W is fresh and $\Phi = \Phi'[W := \sigma' Z'.\Gamma]$. Then $|| \, Z.\Phi \, ||^{\mathcal{T}}_{\mathcal{V}} = f[\sigma']g$, where $f(X,Y) = || \, \Phi' \, ||^{\mathcal{T}}_{\mathcal{V}[Z,W:=X,Y]}$ and $g(X,Y) = || \, \Gamma \, ||^{\mathcal{T}}_{\mathcal{V}[Z,Z':=X,Y]}$.*

Proof. Follows from Lemma 5.14 and the definition of $f[\sigma']g$. □

This result implies that $|| \, \sigma Z.\Phi \, ||^{\mathcal{T}}_{\mathcal{V}} = \sigma(f[\sigma']g)$, where $\sigma Z.\Phi$, f and g are defined as in the lemma.

Quotient well-foundedness and well-orderings. Our goal in this part of the article is to characterize a support structure for g in terms of a given support structure for $f[\sigma]g$. For unary functions, support structures may be either well-founded or not, and this property is sufficient to characterize both the greatest and least fixponts of these functions. For function $f[\sigma]g$, where f and g are binary and have mutually recursive fixpoints, an intermediate notion, which we call *quotient well-foundedness* is important, especially when the mutually recursive fixpoints are of different types (i.e., one is least while the other is greatest). The proofs of the lemmata concerning quotient-well-foundedness and well-orderings are included in the appendix.

Definition 8.4 (Quotient Well-founded (qwf)/Well-Ordering (qwo)). Let S be a set and $R \subseteq S \times S$ a binary relation over S, and let $(\mathcal{Q}_R, \sqsubseteq)$ be the quotient of R (cf. Definition 3.8), with $\sqsubset \, = \, \sqsubseteq^{-}$ the irreflexive core of $\sqsubseteq$.

(1) R is *quotient well-founded* (qwf) iff $\sqsubset$ is well-founded over $\mathcal{Q}_R$.
(2) R is a *quotient well-ordering* (qwo) iff $\sqsubset$ is well-ordering over $\mathcal{Q}_R$.

That is, R is quotient well-founded iff the irreflexive core of the partial order induced by R over its equivalence classes is well-founded. Note that R can be qwf without being well-founded; when this is the case the non-well-foundedness of R can be seen as due solely to non-well-foundedness within its equivalence classes. It is also easy to see that if R is well-founded then it is qwf as well; in this case each $Q \in \mathcal{Q}_R$ has form $\{s\}$ for some $s \in S$. Also note that the universal relation $U_S = S \times S$ over S is trivially qwf, as its quotient has one equivalence class, namely, S.

It turns out that if a relation is total and qwf, then it is also a quotient well-ordering.

Lemma 8.5 (Total Qwf Relations are Qwos). *If $R \subseteq S \times S$ is total and qwf, then R is a qwo.*

The next result establishes the existence of so-called *pseudo-minimum* elements in subsets drawn from qwos. These elements are defined as follows:

Definition 8.6 (Pseudo-minimum Elements). Let $R \subseteq S \times S$ be a binary relation over S, and let $X \subseteq S$. Then $x \in X$ is *R-pseudo-minimum for X* iff x is an R-lower bound for X.

If $x \in X$ is an R-pseudo-minimum for X, this does not imply that x is an R-minimum, or even R-minimal, since even though x is a R-pseudo-minimum there may exist $x' \in X$ such that $x' \, R \, x$. In this case it must hold that $x \sim_R x'$, however.

The next lemma states a pseudo-minimum result for qwo relations that are total. (It should be noted that a relation can be a qwo and still not itself be total.)

Lemma 8.7 (Pseudo-minimum Elements and Quotient Well-orderings). *Suppose qwo $R \subseteq S \times S$ is total. Then every non-empty subset $X \subseteq S$ contains an R-pseudo-minimum element.*

We now establish a relationship between support structures for $f[\sigma]g$ and g. We first define a notion of compatibility for such support structures.

Definition 8.8 (Local Consistency of Support Structures). Let $f, g \in 2^S \times 2^S \to 2^S$ be monotonic, and let $\sigma_1, \sigma_2 \in \{\mu, \nu\}$. Furthermore, let $(X, \prec)$ be a σ_1-compatible support structure for $f[\sigma_2]g$, with $Y_x = \sigma_2 g_{(\prec^{-1}(x),\,\cdot)}$ for $x \in X$ and $Y = \sigma_2\, g_{(\prec^{-1}(X),\,\cdot)}$. Then σ_2-compatible support structure $(Y, \prec')$ for $g_{(\prec^{-1}(X),\,\cdot)}$ is *locally consistent* with $(X, \prec)$ iff for all $x \in X$, $(Y_x, (\prec')\lfloor Y_x)$ is a σ_2-compatible support structure for $g_{(\prec^{-1}(x),\,\cdot)}$ and for all $y \in Y_x$, ${\prec'}^{-1}(y) \subseteq Y_x$.

Intuitively, $(Y, \prec')$ is locally consistent with $(X, \prec)$ if $\prec'$ not only supports the fact that Y is the σ_2-fixpoint for $g_{(\prec^{-1}(X),\,\cdot)}$, but via the restriction of $\prec'$ to Y_x, it also provides localized support for the fact that Y_x is the σ_2-fixpoint for $g_{(\prec^{-1}(x),\,\cdot)}$, for each $x \in X$. In addition, for any $x \in X$ and $y \in Y_x$ every element in the support set ${\prec'}^{-1}(y)$ with respect to $\prec'$ must also be an element of Y_x. Note that this last aspect of the definition ensures that for any x and $y \in Y_x$,

$$ {\prec'}^{-1}(y) = ((\prec')\lfloor Y_x)^{-1}(y). $$

The next lemma establishes that, for given support structures of a specific type for $f[\sigma_2]g$, consistent support structures exist for g. While the statement of this lemma is very different from that of Lemma 4.3, the proofs use very similar constructions.

Lemma 8.9 (From Composite to Local Support Structures). *Let $f, g \in 2^S \times 2^S \to 2^S$ be monotonic and $\sigma_1, \sigma_2 \in \{\mu, \nu\}$, with $X = \sigma_1(f[\sigma_2]g)$. Also let $(X, \prec)$ be a σ_1-compatible, total qwf support structure for $f[\sigma_2]g$ and $Y = \sigma_2 g_{(\prec^{-1}(X),\,\cdot)}$. Then there is a σ_2-compatible, total qwf support structure $(Y, \prec')$ for $g_{(\prec^{-1}(X),\,\cdot)}$ that is locally consistent with $(X, \prec)$.*

8.2 Tableau Normal Form

Later in this section, we use constructions on tableaux to prove completeness of our proof system. The tableaux we work with have a restricted form, which we call *tableau normal form* (TNF). TNF is defined as follows:

Definition 8.10 (TNF). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a tableau, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$. Tableau $\mathbb{T}$ is in TNF iff it satisfies

(1) $\rho(\mathbf{r}) \neq$ Thin and for all $\mathbf{n} \neq \mathbf{r}$, $\rho(\mathbf{n}) = \sigma Z.$ iff $\rho(p(\mathbf{n})) =$ Thin,
(2) for each definitional constant U appearing in $\mathbb{T}$ there is exactly one node $\mathbf{n}_U$ such that $fm(\mathbf{n}_U) = U$ and $\rho(\mathbf{n}_U) =$ Un, and
(3) for each node $\mathbf{n}$ such that $\rho(\mathbf{n}) = \vee$ and $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$, $st(\mathbf{n}_1) \cap st(\mathbf{n}_2) = \emptyset$.

Intuitively, $\mathbb{T}$ is in TNF if Thin is not applied to the root node, every application of the $\sigma Z.$ rule to a non-root node is immediately preceded by a single instance of Thin, and there are otherwise no other applications of Thin; each definitional constant is unfolded exactly once; and for each $\vee$-node, the state sets of the node's children are disjoint (i.e., no state can appear in both children, meaning there can be no redundant reasoning about states in the $\vee$-node).

In what follows we will on occasion build new (successful) TNF tableaux out of existing (successful) TNF tableaux that are *structurally equivalent*.

Definition 8.11 (Structurally Equivalent Tableaux). Tableaux $\mathbb{T}_1 = (\mathbf{T}_1, \rho_1, \mathcal{T}, \mathcal{V}_1, \lambda_1)$ and $\mathbb{T}_2 = (\mathbf{T}_2, \rho_2, \mathcal{T}, \mathcal{V}_2, \lambda_2)$ are *structurally equivalent* if the underlying trees $\mathbf{T}_1$ and $\mathbf{T}_2$ are isomorphic, and for isomorphic nodes $\mathbf{n}_1$ in $\mathbf{T}_1$ and $\mathbf{n}_2$ in $\mathbf{T}_2$ it holds that $rn(\rho_1(\mathbf{n}_1)) = rn(\rho_2(\mathbf{n}_2))$, $fm(\lambda_1(\mathbf{n}_1)) = fm(\lambda_2(\mathbf{n}_2))$, and $dl(\lambda_1(\mathbf{n}_1)) = dl(\lambda_2(\mathbf{n}_2))$. If ι is an isomorphism satisfying these properties, we refer to it as a *structural tableau morphism* from $\mathbb{T}_1$ to $\mathbb{T}_2$.

Two tableaux are structurally equivalent if they are "almost isomorphic", in the standard sense. Specifically, they must involve the same transition system, their trees must be isomorphic, and isomorphic nodes in the two trees must have the same proof rule applied to them, although they may have different witness functions if the rule involved is $\langle K \rangle$. Sequents labeling isomorphic nodes may differ in their valuations, and the set of states mentioned in the sequents, although the formulas and definition lists must be the same. Intuitively, structurally equivalent tableaux may be seen as employing the same reasoning, but on slightly different, albeit similar, sequents.

Call a tableau *diamond-leaf-free* if it contains no diamond leaves; recall that any successful tableau must be diamond-leaf-free. The next lemma establishes that if two diamond-leaf-free TNF tableaux have root sequents $\mathbf{s}_1$ and $\mathbf{s}_2$ such that $fm(\mathbf{s}_1) = fm(\mathbf{s}_2)$ and $dl(\mathbf{s}_1) = dl(\mathbf{s}_2) = \varepsilon$, then the tableaux must be structurally equivalent. It relies on an assumption that we make throughout this section: that definitional constants as introduced in the $\sigma Z.$ rule are generated uniformly. That is, if the sequent labeling a node has form $S \vdash_{\Delta}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi$ and rule $\sigma Z.$ is applied, then a given definitional constant U depending only on $\sigma Z.\Phi$ and Δ is introduced, with the child sequent $S \vdash_{\Delta\cdot(U=\sigma Z.\Phi)}^{\mathcal{T},\mathcal{V}} U$ being generated.

Lemma 8.12 (Structural Equivalence of TNF Tableaux). *Let* $\mathbf{s}_1 = S_1 \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}_1} \Phi$ *and* $\mathbf{s}_2 = S_2 \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}_2} \Phi$ *be sequents, with* $\mathbb{T}_1$ *and* $\mathbb{T}_2$ *diamond-leaf-free TNF tableaux for* $\mathbf{s}_1$ *and* $\mathbf{s}_2$*, respectively. Then* $\mathbb{T}_1$ *and* $\mathbb{T}_2$ *are structurally equivalent.*

Proof. It suffices to give a structural tableau morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$. This can be done co-inductively using $\mathbf{T}_1$ and $\mathbf{T}_2$, the trees embedded in $\mathbb{T}_1$ and $\mathbb{T}_2$. The limitations imposed by TNF on the use of the Thin and Un rules ensure the desired similarities in sequents labeling isomorphic tree nodes, while the diamond-leaf-free property ensures that all leaves must be free leaves, i.e., of form Z or $\neg Z$ for some Z free in Φ, or σ-leaves, i.e., of form U for some U defined in the definition list of the leaf. □

8.3 Completeness via Tableau Constructions

We now turn to proving the existence of successful TNF tableaux for different classes of valid sequents. The first result establishes the existence of such tableaux for valid sequents whose formulas contain no fixpoint subformulas.

Lemma 8.13 (Fixpoint-free Completeness). *Let* $\mathcal{T}, \mathcal{V}, \Phi$ *and* S *be such that* Φ *is fixpoint-free and* $S \subseteq \| \Phi \|_{\mathcal{V}}^{\mathcal{T}}$*. Then there is a successful TNF tableau for* $S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \Phi$.

Proof sketch. Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be an LTS of sort Σ and $\mathcal{V}$ be a valuation. The proof proceeds by structural induction on Φ; the induction hypothesis states for any subformula Φ' of Φ and S' such that $S' \subseteq \| \Phi' \|_{\mathcal{V}}^{\mathcal{T}}$, that $S' \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \Phi'$ has a successful TNF tableau. The argument involves a case analysis on the form of Φ whose cases are routine. □

We now turn to the existence of successful TNF tableaux for different classes of sequents involving fixpoint formulas. We first define the notion of *compliance* between a tableau and a support structure.

Definition 8.14 (Tableau Compliance with $\prec$). Let $\mathcal{T}, \mathcal{V}, Z, \Phi, \sigma$ and S be such that $S = \| \sigma Z.\Phi \|_{\mathcal{V}}^{\mathcal{T}}$. Also let $(S, \prec)$ be a σ-compatible support structure for $\| Z.\Phi \|_{\mathcal{V}}^{\mathcal{T}}$. Then TNF tableau $(\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ with root node $\mathbf{r} = S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi$ and $\mathbf{r}' = cs(\mathbf{r}) = S \vdash_{(U=\sigma Z.\Phi)}^{\mathcal{T},\mathcal{V}} U$ is *compliant* with $\prec$ iff whenever $s' <:_{\mathbf{r}'} s$, then $s' \prec s$.

Intuitively, tableau $\mathbb{T}$ for sequent $S \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \sigma Z.\Phi$ is compliant with support structure $\prec$ iff every extended dependency from $s \in S$ to a state in a companion leaf of $\mathbf{r}'$ is also a semantic dependency as reflected in $\prec$. Note that since the root node of $\mathbb{T}$ is a fixpoint node and $\mathbb{T}$ is in TNF, $\rho(\mathbf{r}) = \sigma Z$.

and thus $cs(\mathbf{r}) = \mathbf{r}' = S \vdash^{\mathcal{T},\mathcal{V}}_{U=\sigma Z.\Phi} U$ for some definitional constant U. Also, since $\mathbb{T}$ is TNF $\rho(\mathbf{r}') =$ Un, and $\mathbf{r}'$ is the only node in which unfolding is applied to the definitional constant U.

The next lemma continues the sequence of results in this section on the existence of successful TNF tableaux for valid sequents. In this case, formulas have form $\sigma Z.\Phi$, where Φ contains no fixpoint subformulas, and have specific σ-compatible support structures, and the result shows how successful TNF tableaux that are compliant with the given support structure can be constructed.

Lemma 8.15 (Single-fixpoint Completeness). *Fix $\mathcal{T}$, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that Φ is fixpoint-free and $S = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful TNF tableau compliant with $(S, \prec)$.*

Proof. Fix $\mathcal{T} = (\mathcal{S}, \rightarrow)$ of sort Σ, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that Φ is fixpoint-free and $S = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $f = ||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. We must construct a successful TNF tableau for sequent $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ that is compliant with $(S, \prec)$. The proof consists of the following steps.

(1) For each $s \in S$ we use Lemma 8.13 to establish the existence of a successful TNF tableau for sequent $\{s\} \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$, where $\mathcal{V}_s = \mathcal{V}[Z := \prec^{-1}(s)]$.
(2) We then construct a successful TNF tableau for sequent $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$, where $\mathcal{V}_S = \mathcal{V}[Z := \prec^{-1}(S)]$, from the individual tableaux for the $s \in S$.
(3) We convert the tableau for $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$ into a successful TNF tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ that is compliant with $\prec$.

Step 1 of proof outline: construct tableau for $\{s\} \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$, where $s \in S$. For any $s \in S$ we have that $s \in ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}_s}$, meaning $\{s\} \subseteq ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$ is valid. Since Φ is fixpoint-free Lemma 8.13 guarantees the existence of a successful TNF tableau

$$\begin{aligned} \mathbb{T}_s &= (\mathbf{T}_s, \rho_s, \mathcal{T}, \mathcal{V}_s, \lambda_s), \text{where} \\ \mathbf{T}_s &= (\mathbf{N}_s, \mathbf{r}_s, p_s, cs_s) \end{aligned}$$

for $\{s\} \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$. We now remark on some properties of $\mathbb{T}_s$.

(1) Suppose s' and $\mathbf{n}' \in \mathbf{N}_s$ are such that $fm(\mathbf{n}') = Z$ (so $\mathbf{n}'$ is a leaf whose formula is Z, the variable bound by σ in $\sigma Z.\Phi$) and $s' <:_{\mathbf{n}',\mathbf{r}_s} s$. Then $s' \prec s$, since $s' \in \mathcal{V}_s(Z) = \prec^{-1}(s)$.
(2) Let $s' \in S$, and let $\mathbb{T}_{s'}$ be the successful TNF tableau for $\{s'\} \vdash^{\mathcal{T},\mathcal{V}_{s'}}_{\varepsilon} \Phi$. Lemma 8.12 and the fact that successful tableaux must be diamond-leaf-free guarantee that $\mathbb{T}_s$ and $\mathbb{T}_{s'}$ are structurally equivalent.

Observation 1 establishes a relationship between dependencies involving the single state in the root node of $\mathbb{T}_s$ and states in the leaves involving Z. Observation 2 guarantees that the successful TNF tableaux due to Lemma 8.13 are structurally equivalent, and hence isomorphic as trees, and satisfying the property that isomorphic nodes in these trees share the same formulas, definition lists (ε in this case) and rule names. In what follows, we use $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, $fm(\mathbf{n})$ and $rn(\mathbf{n})$ for these common structures and write $\mathbb{T}_s = (\mathbf{T}, \rho_s, \mathcal{T}, \mathcal{V}_s, \lambda_s)$ for $s \in S$, noting that for all $s, s' \in S$, $rn(\rho_s(\mathbf{n})) = rn(\rho_{s'}(\mathbf{n})) = rn(\mathbf{n})$.

Step 2 of proof outline: construct tableau for $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$. We now construct a successful TNF tableau for $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$ satisfying the following: if s, s' and $\mathbf{n}'$ are such that $fm(\mathbf{n}') = Z$ and $s' <:_{\mathbf{n}',\mathbf{r}} s$, then $s' \prec s$. There are two cases to consider. In the first case, $S = \emptyset$. In this case, $\prec = \prec^{-1}(S) = \emptyset$, and $\emptyset \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$ is valid and therefore, by Lemma 8.13, has a successful TNF tableau. Define $\mathbb{T}_S$ to be this tableau. Note that, since $S = \emptyset$, T_S vacuously satisfies the property involving $<:$.

In the second case, $S \neq \emptyset$; we will construct $\mathbb{T}_S = (\mathbf{T}, \rho_S, \mathcal{T}, \mathcal{V}_S, \lambda_S)$ that is structurally equivalent to each $\mathbb{T}_s$ for $s \in S$. The intuition behind the construction is to "merge" the individual tableaux $\mathbb{T}_s$ for the $s \in S$ by assigning to each node $\mathbf{n}$ in $\mathbb{T}_S$ the set of states obtained by appropriately combining all the sets of states each individual tableau $\mathbb{T}_s$ assigns to the node. Care must be taken with nodes involving the $\vee$ and $\langle K \rangle$ proof rules.

Since $\mathbf{T}$ is already given, completing the construction of $\mathbb{T}_S$ only requires that we define ρ_S and λ_S, which we do so that the following invariants hold for each $\mathbf{n} \in \mathbf{N}$.

I1. If $\rho_S(\mathbf{n})$ is defined, then $rn(\rho_S(\mathbf{n})) = rn(\mathbf{n})$ and the sequents assigned by λ_S to $\mathbf{n}$ and its children are consistent with $\rho_S(\mathbf{n})$.
I2. $fm(\lambda_S(\mathbf{n})) = fm(\mathbf{n})$.
I3. $st(\lambda_S(\mathbf{n})) \subseteq \bigcup_{s \in S} st(\lambda_s(\mathbf{n}))$.

The definitions of ρ_S and λ_S are given in a co-inductive fashion (i.e., "from the root down"). We begin by taking $\lambda_S(\mathbf{r}) = S \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} \Phi$; invariants I2 and I3 clearly hold of $\lambda_S(\mathbf{r})$. For the co-inductive step we assume that $\mathbf{n}$ satisfies I2 and I3 and define $\rho_S(\mathbf{n}')$ and $\lambda(\mathbf{n}')$ for each child $\mathbf{n}'$ of $\mathbf{n}$ so that I1 holds of $\mathbf{n}$ and I2 and I3 hold of each of the $\mathbf{n}'$. This is done below based on $rn(\mathbf{n})$, the name of the rule applied to $\mathbf{n}$. Note that because each $\mathbb{T}_s$ is in TNF and Φ is fixpoint-free there can be no $\mathbf{n} \in \mathbf{N}$ such that $rn(\mathbf{n}) \in \{\text{Thin}, \sigma Z, \text{Un}\}$. In what follows we let $S_{\mathbf{n}} = st(\lambda_S(\mathbf{n}))$ be the set of states in the sequent labeling $\mathbf{n}$. Most cases are left to the reader; we detail only the cases when $rn(\mathbf{n})\bot$ and $rn(\mathbf{n}) = \langle K \rangle$.

***rn*(n)⊥.** In this case, I1 holds vacuously. Note that $\mathbf{n}$ must be a leaf and thus has no children.

***rn*(n) = ⟨*K*⟩.** In this case, $cs(\mathbf{n}) = \mathbf{n}'$ and $fm(\mathbf{n}) = \langle K \rangle fm(\mathbf{n}')$. To define $\rho_S(\mathbf{n})$ we first construct a witness function $f_{\mathbf{n}} \in S_{\mathbf{n}} \to \mathcal{S}$ such that $s \xrightarrow{K} f_{\mathbf{n}}(s)$ for all $s \in S_{\mathbf{n}}$ and such that $f_{\mathbf{n}}(S_{\mathbf{n}}) \subseteq \bigcup_{s \in S} st(\lambda_s(\mathbf{n}'))$. This function will then be used to define the sequent labeling $\mathbf{n}'$. So fix $s \in S_{\mathbf{n}}$; we construct $f_{\mathbf{n}}(s)$ based on the tableaux $\mathbb{T}_{s'}$ whose sequent for $\mathbf{n}$ contains s. To this end, define

$$I_s = \{s' \in S \mid s \in st(\lambda_{s'}(\mathbf{n}))\}.$$

Intuitively, $I_s \subseteq S$ contains all states s' whose tableau $\mathbb{T}_{s'}$ contains state s in $\mathbf{n}$. Clearly I_s is non-empty and thus contains a pseudo-minimum element s' w.r.t. $\prec$ (Lemma 8.7). Now consider $\rho_{s'}(\mathbf{n})$; it has form $(\langle K \rangle, f_{\mathbf{n},s'})$, where $f_{\mathbf{n},s'}$ is the witness function for $\mathbf{n}$ in tableau $\mathbb{T}_{s'}$. This means that $f_{\mathbf{n},s'} \in st(\lambda_{s'}(\mathbf{n})) \to \mathcal{S}$ is such that $st(\lambda_{s'}(\mathbf{n}')) = f_{\mathbf{n},s'}(st(\lambda_{s'}(\mathbf{n})))$. We now define $f_{\mathbf{n}}(s) = f_{\mathbf{n},s'}(s)$, $\rho_S(\mathbf{n}) = (\langle K \rangle, f_{\mathbf{n}})$, and $\lambda_S(\mathbf{n}') = f_{\mathbf{n}}(S_{\mathbf{n}}) \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} fm(\mathbf{n}')$. It can be seen that invariant I1 holds of $\mathbf{n}$ and that I2 and I3 hold of $\mathbf{n}'$.

This construction ensures that Properties I1–I3 hold for all $\mathbf{n}$.

To establish that $\mathbb{T}_S$ is successful, we must show that every leaf in $\mathbb{T}_S$ is successful (cf. Definition 6.13), which amounts to showing that for each such leaf $\mathbf{n}$, $st(\lambda_S(\mathbf{n})) \subseteq ||\, fm(\mathbf{n})\, ||_{\mathcal{V}_S}^{\mathcal{T}}$. Since Φ contains no fixpoint subformulas there are two cases to consider.

***fm*(n) = *Z*.** In this case, the formula labeling $\mathbf{n}$ is the bound variable Z in $\sigma Z.\Phi$. Since, for each $s \in S$, $\mathbb{T}_s$ is successful and $\prec^{-1}(s) \subseteq \prec^{-1}(S)$, we have

$$st(\lambda_s(\mathbf{n})) \subseteq ||\, fm(\mathbf{n})\, ||_{\mathcal{V}_s}^{\mathcal{T}} = \mathcal{V}_s(Z) = \prec^{-1}(s).$$

As invariant I3 ensures that $st(\lambda_S(\mathbf{n})) \subseteq \bigcup_{s \in S} st(\lambda_s(\mathbf{n}))$, it follows that

$$st(\lambda_S(\mathbf{n})) \subseteq \bigcup_{s \in S} st(\lambda_s(\mathbf{n})) = \bigcup_{s \in S} \prec^{-1}(s) = \prec^{-1}(S) = \mathcal{V}_S(Z) = ||\, fm(\mathbf{n})\, ||_{\mathcal{V}_S}^{\mathcal{T}}.$$

Leaf $\mathbf{n}$ is, therefore, successful.

***fm*(n) $\in \{Y, \neg Y\}$ for some $Y \neq Z$ free in $\sigma Z.\Phi$.** The argument is very similar to the previous case; the only difference is that for any $s \in S$, $\mathcal{V}_s(Y) = \mathcal{V}_S(Y)$ and thus, $||\,fm(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}_s} = ||\,fm(\mathbf{n})\,||^{\mathcal{T}}_{\mathcal{V}_S}$.

That $\mathbb{T}_S$ is in TNF follows from the fact that it is successful (and thus, diamond-leaf-free) and that each $\mathbb{T}_s$ is successful and TNF, and from the definitions of ρ_S and λ_S.

We now establish that for all leaves $\mathbf{n}$ in $\mathbb{T}_S$ such that $fm(\mathbf{n}) = Z$, and $s_\mathbf{n}, s_\mathbf{r} \in S$ such that $s_\mathbf{n} <:_{\mathbf{n},\mathbf{r}} s_\mathbf{r}$ in $\mathbb{T}_S$, $s_\mathbf{n} \prec s_\mathbf{r}$. We begin by noting that if $\mathbf{n} = \mathbf{r}$ then $\Phi = Z$ and $s_\mathbf{n} <:_{\mathbf{n},\mathbf{r}} s_\mathbf{r}$ iff $s_\mathbf{n} = s_\mathbf{r}$. In this case, if $\sigma = \nu$ then $S = ||\,\nu Z.Z\,||^{\mathcal{T}}_{\mathcal{V}} = \mathcal{S}$ and the result holds because $(S, \prec)$ is a support structure for f, and thus must be reflexive since f is the identity function. If instead $\sigma = \mu$ then $S = ||\,\mu Z.Z\,||^{\mathcal{T}}_{\mathcal{V}} = \emptyset$ and the result is vacuously true.

Now assume that $\mathbf{n} \neq \mathbf{r}$. We start by remarking on a property that holds of all $\mathbf{n}_1, \mathbf{n}_2, s_1$ and s_2 such that $s_2 <_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}_S$: for every $s \in S$ such that $s_1 \in st(\lambda_s(\mathbf{n}_1))$, either $s_2 \in st(\lambda_s(\mathbf{n}_2))$, or there exists $s' \prec s$ such that $s_2 \in st(\lambda_{s'}(\mathbf{n}_2))$. That is, when $s_2 <_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}_S$, meaning $\mathbf{n}_2$ is a child of $\mathbf{n}_1$ and s_2 is a direct dependent of s_1 in $\mathbb{T}_S$, and $\mathbb{T}_s$ is a tableau containing s_1 in $\mathbf{n}_1$, then s_2 is also contained in $\mathbf{n}_2$ of either $\mathbb{T}_s$ or $\mathbb{T}_{s'}$ for some $s' \prec s$. This is easily observed based on the definition of $<_{\mathbf{n}_1,\mathbf{n}_2}$, as well as the construction of $\mathbb{T}_S$ above and its use of pseudo-minimal states in the $\vee$ and $\langle K \rangle$ cases. A simple inductive argument lifts this result to the case when $\mathbf{n}_1 \neq \mathbf{n}_2$ and s_1, s_2 are such that $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}_S$: for all $s \in S$ such that $s_1 \in st(\lambda_s(\mathbf{n}_1))$, either $s_2 \in st(\lambda_s(\mathbf{n}_2))$ or there exists $s' \prec s$ such that $s_2 \in st(\lambda_{s'}(\mathbf{n}_2))$. Now consider $\mathbf{n}$ and $\mathbf{r}$ as given above, and assume $s_\mathbf{n} <:_{\mathbf{n},\mathbf{r}} s_\mathbf{r}$. It follows from the definition of $\mathbb{T}_s$ that if $s_\mathbf{r} \in st(\lambda_s(\mathbf{r}))$ then $s = s_\mathbf{r}$, since $st(\lambda_s(\mathbf{r})) = \{s\}$. There are now two cases to consider. In the first, $s_\mathbf{n} \in st(\lambda_s(\mathbf{n}))$, whence $s_\mathbf{n} \in \mathcal{V}_s(Z)$ and $s_\mathbf{n} \prec s = s_\mathbf{r}$. In the second, there is an $s' \in S$ such that $s' \prec s_\mathbf{r}$ and $s_\mathbf{n} \in st(\lambda_{s'}(\mathbf{n}))$. In this case, $s_\mathbf{n} \in \mathcal{V}_{s'}(Z) = \prec^{-1}(s')$, meaning that $s_\mathbf{n} \prec s'$. Since $\prec$ is total, and hence transitive, we have $s_\mathbf{n} \prec s' \prec s_\mathbf{r}$ and thus $s_\mathbf{n} \prec s_\mathbf{r}$.

Step 3 of proof outline: construct tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$. To complete the proof, we convert $\mathbb{T}_S$ into a tableau $\mathbb{T}_\sigma = (\mathbf{T}_\sigma, \rho_\sigma, \mathcal{T}, \mathcal{V}, \lambda_\sigma)$ for sequent $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ as follows. We create two fresh tree nodes $\mathbf{r}_1, \mathbf{r}_2 \notin \mathbf{N}$ and add these into $\mathbf{T}_\sigma$ along with all the nodes of $\mathbf{T}$. The root of $\mathbf{T}_\sigma$ is taken to be $\mathbf{r}_1$; the parent of $\mathbf{r}_2$ is then $\mathbf{r}_1$, while the parent of $\mathbf{r}$, the original root of $\mathbf{T}$, is $\mathbf{r}_2$. The other nodes of $\mathbf{T}$ retain their parents and sibling structure from $\mathbf{T}$. We also define $\rho_\sigma(\mathbf{r}_1) = \sigma Z$ and $\rho_\sigma(\mathbf{r}_2) = \text{Un}$; for all nodes $\mathbf{n}$ in $\mathbf{T}$, $\rho_\sigma(\mathbf{n}) = \rho_S(\mathbf{n})$. We now define λ_σ as follows:

$$\lambda_\sigma(\mathbf{n}) = \begin{cases} S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi & \text{if } \mathbf{n} = \mathbf{r}_1 \\ S \vdash^{\mathcal{T},\mathcal{V}}_{(U=\sigma Z.\Phi)} U & \text{if } \mathbf{n} = \mathbf{r}_2, U \text{ fresh} \\ S' \vdash^{\mathcal{T},\mathcal{V}}_{(U=\sigma Z.\Phi)} \Phi'[Z := U] & \text{if } \lambda_S(\mathbf{n}) = S' \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi'. \end{cases}$$

To finish the proof we must establish that $\mathbb{T}_\sigma$ is a successful TNF tableau compliant with $(S, \prec)$. That $\mathbb{T}_\sigma$ is a tableau follows from the fact that $\mathbb{T}_S$ is a successful tableau. Now consider σ-leaf $\mathbf{n}$ in $\mathbb{T}_\sigma$; that is, $\lambda_\sigma(\mathbf{n}) = S_\mathbf{n} \vdash^{\mathcal{T},\mathcal{V}}_{(U=\sigma Z.\Phi)} U$. Since $\mathbb{T}_S$ is successful and $\lambda_S(\mathbf{n}) = S_\mathbf{n} \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} Z$, $S_\mathbf{n} \subseteq \mathcal{V}_S(Z) = \prec^{-1}(S) \subseteq S$. Since $\mathbf{r}_2$ is the only node with an application of rule Un, it must be the companion node of $\mathbf{n}$ in $\mathbb{T}_\sigma$. Since $st(\lambda_\sigma(\mathbf{r}_2)) = S$, $\mathbf{n}$ is terminal, and $\mathbb{T}_\sigma$ is indeed a tableau.

The fact that $\mathbb{T}_S$ is TNF and that U is unfolded only once in $\mathbb{T}_\sigma$ guarantees that $\mathbb{T}_\sigma$ is TNF.

We now argue that $\mathbb{T}_\sigma$ is compliant with $(S, \prec)$. To this end, suppose that $s_1, s_2 \in S$ are such that $s_2 <:_{\mathbf{r}_2} s_1$ in $\mathbb{T}_\sigma$; we must show that $s_2 \prec s_1$. This follows immediately from the fact that $s_2 <:_{\mathbf{r}_2} s_1$ in $\mathbb{T}_\sigma$ iff there is a leaf $\mathbf{n}$ in $\mathbb{T}_S$ such that $fm(\lambda_S(\mathbf{n})) = Z$, $s_2 \in st(\lambda_S(\mathbf{n}))$, and $s_2 <:_{\mathbf{n},\mathbf{r}} s_1$ in $\mathbb{T}_S$. Previous arguments then establish that $s_2 \prec s_1$.

To establish that $\mathbb{T}_\sigma$ is successful we must show that each of its leaves is successful. Suppose $\mathbf{n}$ is a non-σ-leaf leaf; that is, $fm(\lambda_\sigma(\mathbf{n})) \neq U$. In this case $\lambda_\sigma(\mathbf{n}) = \lambda_S(\mathbf{n})$, and the success of $\mathbb{T}_S$

and all its leaves guarantees the success of this leaf. Now suppose that $\mathbf{n}$ is a σ-leaf, meaning that $fm(\lambda_\sigma(\mathbf{n})) = U$. If $\sigma = \nu$ then this leaf is successful. If $\sigma = \mu$ then we note that for $(S, \prec)$ to be a support structure for $S = \mu f$, $\prec$ must be well-founded. Compliance of $\mathbb{T}_\sigma$ with $\prec$ guarantees that $<:_{\mathbf{r}_2}$ is also well-founded, and thus $\mathbf{n}$ is successful in this case also. This completes the proof. □

As an immediate corollary, we have the following:

Corollary 8.16. *Fix $\mathcal{T}$, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that Φ is fixpoint-free and $S = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful tableau.*

Proof. Follows from Lemma 8.15, the fact that every σ-maximal support structure $(S, \prec)$ for $||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$ is total and qwf, and the fact that Corollary 4.11 guarantees the existence of such σ-maximal support structures. □

We now state and prove a generalization of Lemma 8.15 in which the body of the fixpoint formula is allowed also to have fixpoint subformulas.

Lemma 8.17 (Fixpoint Completeness). *Fix $\mathcal{T}$, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that $S = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful TNF tableau compliant with $(S, \prec)$.*

Proof. Fix $\mathcal{T} = (\mathcal{S}, \rightarrow)$ of sort Σ. We prove the following: for all Φ, and $Z, \mathcal{V}, \sigma$ and $(S, \prec)$, if $S = ||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$, and $(S, \prec)$ is a σ-compatible, total qwf support structure for $||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$, then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful TNF tableau $\mathbb{T}_\Phi$ that is compliant with $(S, \prec)$. To simplify notation we use the following abbreviations:

$$f_\Phi = ||\,Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}},$$
$$\mathcal{V}_X = \mathcal{V}[Z := \prec^{-1}(X)].$$

Note that $\mathcal{V}_S = \mathcal{V}[Z := \prec^{-1}(S)]$. When $s \in S$ we also write $\mathcal{V}_s$ in lieu of $V_{\{s\}}$.

The proof proceeds by strong induction on the number of fixpoint subformulas of Φ. There are two cases to consider. In the first case, Φ contains no fixpoint formulas. Lemma 8.15 immediately gives the desired result.

In the second case, Φ contains at least one fixpoint subformula. The outline of the proof in this case is as follows:

(1) We decompose Φ into Φ', which uses a new free variable W, and $\sigma' Z'.\Gamma$ in such a way that $\Phi = \Phi'[W := \sigma' Z'.\Gamma]$.
(2) We inductively construct a successful TNF tableau $\mathbb{T}_{\Phi'}$ for $S \vdash^{\mathcal{T},\mathcal{V}'}_{\varepsilon} \sigma Z.\Phi'$ that is compliant with $(S, \prec)$, where $\mathcal{V}' = \mathcal{V}[W := S']$ and $S' = ||\,\sigma' Z'.\Gamma\,||^{\mathcal{T}}_{\mathcal{V}_S}$. ($S'$ may be seen as the semantic content of $\sigma' Z'.\Gamma$ relevant for $||\,\sigma Z.\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$.)
(3) We construct a successful TNF tableau $\mathbb{T}_\Gamma$ satisfying a compliance-related property for $S' \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \sigma' Z'.\Gamma$.
(4) We show how to compose $\mathbb{T}_\Phi$ and $\mathbb{T}_\Gamma$ to yield a successful TNF tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ that is compliant with $(S, \prec)$.

We now work through each of these proof steps.

Step 1 of proof outline: decompose Φ. Let $\sigma' Z'.\Gamma$ be a maximal fixpoint subformula in Φ as defined previously in this section. Also let $W \in \text{Var}$ be a fresh propositional variable, and define Φ' so that it contains exactly one instance of W and so that

$$\Phi = \Phi'[W := \sigma' Z'.\Gamma]$$

(Φ' is obtained by replacing one maximal instance of $\sigma' Z'.\Gamma$ in Φ by W.) Note that Φ' and Γ contain strictly fewer fixpoint subformulas than Φ. Lemma 8.3 may now be applied to conclude that $f_\Phi = f[\sigma']g$, where $f, g \in 2^{\mathcal{S}} \times 2^{\mathcal{S}} \to 2^{\mathcal{S}}$ are defined as follows:

$$f(X, Y) = ||\, \Phi' \,||^{\mathcal{T}}_{\mathcal{V}[Z, W := X, Y]},$$
$$g(X, Y) = ||\, \Gamma \,||^{\mathcal{T}}_{\mathcal{V}[Z, Z' := X, Y]}.$$

It is the case that $(S, \prec)$ is a σ-compatible, total qwf support structure for $f[\sigma']g$ since it is for f_Φ and $f_\Phi = f[\sigma']g$.

Step 2 of proof outline: construct tableau $\mathbb{T}_{\Phi'}$ for $S \vdash^{\mathcal{T}, \mathcal{V}'}_{\varepsilon} \sigma Z.\Phi'$. Since Φ' has strictly fewer fixpoint subformulas than Φ, we wish to apply the induction hypothesis to infer the existence of successful TNF tableau $\mathbb{T}_{\Phi'}$ for $S \vdash^{\mathcal{T}, \mathcal{V}'}_{\varepsilon} \sigma Z.\Phi'$ that is compliant with $(S, \prec)$. To do this it suffices to confirm that $S = ||\, \sigma Z.\Phi' \,||^{\mathcal{T}}_{\mathcal{V}'} = \sigma f_{\Phi'}$, where $f_{\Phi'} = ||\, Z.\Phi' \,||^{\mathcal{T}}_{\mathcal{V}'}$, and that $(S, \prec)$ is a support structure for $f_{\Phi'}$ (it is already σ-compatible, total and qwf). That these results hold is left to the reader. We may thus apply the induction hypothesis to infer the existence of successful TNF tableau

$$\mathbb{T}_{\Phi'} = (\mathbf{T}_{\Phi'}, \rho_{\Phi'}, \mathcal{T}, \mathcal{V}', \lambda_{\Phi'}),$$

where $\mathbf{T}_{\Phi'} = (\mathbf{N}_{\Phi'}, \mathbf{r}_{\Phi'}, p_{\Phi'}, cs_{\Phi'})$, such that $\lambda_{\Phi'}(\mathbf{r}_{\Phi'}) = S \vdash^{\mathcal{T}, \mathcal{V}'}_{\varepsilon} \sigma Z.\Phi'$ and $\mathbb{T}_{\Phi'}$ is compliant with $(S, \prec)$. Note that $\mathbb{T}_{\Phi'}$ contains exactly one successful leaf $\mathbf{n}_W$ such that $fm(\lambda_{\Phi'}(\mathbf{n}_W)) = W$.

Step 3 of proof outline: construct tableau $\mathbb{T}_\Gamma$ for $S' \vdash^{\mathcal{T}, \mathcal{V}_S}_{\varepsilon} \sigma' Z'.\Gamma$. In this part of the proof we construct a successful TNF tableau for $S' \vdash^{\mathcal{T}, \mathcal{V}_S}_{\varepsilon} \sigma' Z'.\Gamma$. This tableau is intended to be merged into the tableau $\mathbb{T}_{\Phi'}$ presented above to yield a successful tableau $\mathbb{T}_\Phi$ for sequent $S \vdash^{\mathcal{T}, \mathcal{V}}_{\varepsilon} \sigma Z.\Phi$. The success of this composite tableau $\mathbb{T}_\Phi$ will depend in part on extended dependencies from states in its root sequent, which is $S \vdash^{\mathcal{T}, \mathcal{V}}_{\varepsilon} \sigma Z.\Phi$, to states in the leaves of $\mathbb{T}_\Gamma$ whose formula is Z, the variable bound in $\sigma Z.\Phi$. For this reason, our construction needs to enforce an additional property, which we call (Γ, Φ)*-conformance*, besides success and TNF on $\mathbb{T}_\Gamma$. (Γ, Φ)-conformance is a property of successful, TNF tableaux whose root formula is $\sigma' Z'.\Gamma$; it imposes constraints on states in the Z-leaves of the tableau and states in the root sequent. Intuitively, if x is a state in the root sequent of $\mathbb{T}_\Phi$, it, and any state in its equivalence class $[x]$, is supported by a subset of the states in the root sequent $\mathbf{r}$ of $\mathbb{T}_\Gamma$, denoted $g_{[x]}$. These may in turn depend on other states in the fixpoint $\sigma' g_{[x]}$, which we denote below as $S'_{[x]}$. The definition of (Γ, Φ)-conformance guarantees that, if a state $y \in S'_{[x]}$ that is also in $\mathbf{r}$ syntactically depends on x' in a leaf $\mathbf{n}'$ labeled with variable Z corresponding to the root of $\mathbb{T}_\Phi$, that is, $x' <:_{\mathbf{n}', \mathbf{r}}$ then there also is a semantic dependency of x on x', i.e., $x' \prec x$. In case of a least fixed point, well-foundedness of $<:$ now follows from well-foundedness of $\prec$.

Defining (Γ, Φ)-conformance requires some auxiliary notions, which we introduce here. Suppose that $X \subseteq S$. Then we define

$$g_X = g_{(\prec^{-1}(X), \cdot)},$$
$$S'_X = \sigma' g_X.$$

That is, $g_X \in 2^{\mathcal{S}} \to 2^{\mathcal{S}}$ computes the semantics of Γ, which has both Z and Z' free, with the semantics of Z fixed to be $\prec^{-1}(X)$ and Z' interpreted as the input provided to g_X. S'_X is then the σ' fixpoint of g_X. It can easily be seen that for any $X \subseteq S$, $S'_X \subseteq S'$.

Now let partial order $(\mathcal{Q}_\prec, \sqsubseteq)$ be the quotient of $(S, \prec)$ (cf. Definition 3.8), with $\sqsubset$ the irreflexive core of $\sqsubseteq$. The totality of $\prec$ guarantees that if $Q_1 \sqsubseteq Q_2$ then $\prec^{-1}(Q_1) \subseteq \prec^{-1}(Q_2)$. If $x \in S$ then we write $[x] \in \mathcal{Q}_\prec$ for the equivalence class of x. Since $\prec$ is total it follows that for all x, x', if $x' \in [x]$ then also $x \in [x']$ and $\prec^{-1}(x) = \prec^{-1}(x')$.

We can now define (Γ, Φ)-conformance as follows. Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}', \lambda)$, where $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$, be a successful TNF tableau whose root sequent has form $S'' \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}'} \sigma' Z'.\Gamma$ for some valuation $\mathcal{V}'$. Then $\mathbb{T}$ is (Γ, Φ)-*conformant* iff for all $x \in S$, $y \in S'' \cap S'_{[x]}$, $\mathbf{n}' \in \mathbf{N}$, if $fm(\lambda(\mathbf{n}')) = Z$, and $x' <:_{\mathbf{n}', \mathbf{r}} y$, then $x' \prec x$ holds. In effect, this property asserts a correspondence between $\prec$ and the extended-dependency relation $<:$ in $\mathbb{T}$.

We now define our construction of successful, TNF, compliant, (Γ, Φ)-conformant $\mathbb{T}_\Gamma$. There are two cases to consider. In the first, $S' = \emptyset$. Since Γ contains strictly fewer fixpoint subformulas than Φ we may use the induction hypothesis to conclude that there exists a successful, TNF, $\emptyset$-compliant tableau $\mathbb{T}_\Gamma$ for $S' \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} \sigma' Z'.\Gamma$. (Note that since $S' = \emptyset$, structure $(S', \emptyset)$ is the only support structure for $|| \, Z'.\Gamma \, ||_{\mathcal{V}_S}^{\mathcal{T}}$.) As $S' = \emptyset$, tableau $\mathbb{T}_\Gamma$ is also vacuously (Γ, Φ)-conformant.

In the second case $S' \neq \emptyset$. As in the previous case we may apply the induction hypothesis to construct a successful, TNF tableau for $S' \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} \sigma' Z'.\Gamma$ compatible with an appropriately chosen support structure for $|| \, Z'.\Gamma \, ||_{\mathcal{V}_S}^{\mathcal{T}}$. However, this tableau is *not* guaranteed to be (Γ, Φ)-conformant, and a different construction must be used to ensure this additional property holds. The remainder of this part of the proof is devoted to defining this construction and arguing for its correctness.

To begin with, in order to apply the induction hypothesis to generate successful TNF compliant tableaux for sequents involving $\sigma' Z'.\Gamma$ we also need σ'-compatible, total qwf support structures for the values of g_X we wish to consider. We obtain these from Lemma 8.9 as follows. Recall that $(S, \prec)$ is a σ-compatible total qwf support structure for f_Φ, and that $f_\Phi = f[\sigma']g$. The lemma guarantees the existence of a σ'-compatible, total qwf support structure $(S', \prec')$ for g_S that is locally consistent[9] with $(S, \prec)$.

The construction we present below for $\mathbb{T}_\Gamma$ now proceeds in three steps.

— For each $Q \in \boldsymbol{Q}_\prec$ we inductively construct a successful TNF tableau $\mathbb{T}_{\Gamma, Q}$ for sequent $S'_Q \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_Q} \sigma' Z'.\Gamma$ that is compliant with a subrelation of $\prec'$ and is also (Γ, Φ)-conformant.

— We then merge the individual $\mathbb{T}_{\Gamma, Q}$ to form a successful TNF tableau $\mathbb{T}'_\Gamma$ that is compliant with $\prec'$ and (Γ, Φ)-conformant, and whose root-sequent state set is the union of all the root-sequent state sets of the $\mathbb{T}_{\Gamma, Q}$.

— We then merge $\mathbb{T}'_\Gamma$ with an inductively constructed successful, TNF tableau for $S' \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} \sigma' Z'.\Gamma$ in such a way that the resulting tableau is successful, TNF, compliant with $(S', \prec')$ and (Γ, Φ)-conformant. We take this tableau as $\mathbb{T}_\Gamma$.

Constructing $\mathbb{T}_{\Gamma, Q}$. We begin by noting that since $(S', \prec')$ is locally consistent with $(S, \prec)$ and $\prec^{-1}(x) = \prec^{-1}(x')$ if $x \in [x']$ it follows that for any $Q \in \boldsymbol{Q}_\prec$, $(S'_Q, \prec'_Q)$, where $\prec'_Q = \prec' \lfloor S'_Q$, is a σ'-compatible, total qwf support structure for g_Q. Since Γ contains strictly fewer fixpoint subformulas than Φ, the induction hypothesis guarantees, for each $Q \in \boldsymbol{Q}_\prec$, the existence of a successful TNF tableau

$$\mathbb{T}_{\Gamma, Q} = (\mathbf{T}_{\Gamma, Q}, \rho_{\Gamma, Q}, \mathcal{T}, \mathcal{V}_Q, \lambda_{\Gamma, Q})$$

for $S'_Q \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_Q} \sigma' Z'.\Gamma$, where $\mathbf{T}_{\Gamma, Q} = (\mathbf{N}_{\Gamma, Q}, \mathbf{r}_{\Gamma, Q}, p_{\Gamma, Q}, cs_{\Gamma, Q})$, that is compliant with $\prec'_Q$.

We now note that Lemma 8.12 guarantees that for all $Q, Q' \in \boldsymbol{Q}_\prec$, $\mathbb{T}_{\Gamma, Q}$ and $\mathbb{T}_{\Gamma, Q'}$ are structurally equivalent. In other words, the only differences between these tableaux are the state sets appearing in the sequents at each tree node and the witness functions used in rule applications involving $\langle K \rangle$. We consequently assume in what follows that there is a single common tree $\mathbf{T}_\Gamma = (\mathbf{N}_\Gamma, \mathbf{r}_\Gamma, p_\Gamma, cs_\Gamma)$ for all the $\mathbb{T}_{\Gamma, Q}$. We also introduce the following functions with respect to $\mathbf{N}$ that return the

[9]See Definition 8.8 for the meaning of local consistency.

common elements in the sequents and rule applications labeling the tree nodes.

$fm_\Gamma(\mathbf{n})$ – the formula labeling $\mathbf{n}$ in all the $\mathbb{T}_{\Gamma,Q}$.
$dl_\Gamma(\mathbf{n})$ – the definition list labeling $\mathbf{n}$ in all the $\mathbb{T}_{\Gamma,Q}$.
$rn_\Gamma(\mathbf{n})$ – the rule name in the rule application for $\mathbf{n}$ in all the $\mathbb{T}_{\Gamma,Q}$.

It is easy to verify that each $\mathbb{T}_{\Gamma,Q}$ is also (Γ, Φ)-conformant, as every state x' in a Z-leaf of $\mathbb{T}_{\Gamma,Q}$ by construction satisfies $x' \prec x$ for every $x \in Q = [x]$.

Constructing $\mathbb{T}'_\Gamma$. Since $S \neq \emptyset$, also $\mathcal{Q}_\prec \neq \emptyset$, and we can merge the non-empty set of tableaux $\{\mathbb{T}_{\Gamma,Q} \mid Q \in \mathcal{Q}_\prec\}$ into a single successful tableau $\mathbb{T}'_\Gamma = (\mathbf{T}_\Gamma, \rho'_\Gamma, \mathcal{T}, \mathcal{V}_S, \lambda'_\Gamma)$, sharing the same tree $\mathbf{T}_\Gamma$ and functions rn_Γ, fm_Γ, and dl_Γ as the $\mathbb{T}_{\Gamma,Q}$, with the following properties:

G1. $\lambda'_\Gamma(\mathbf{r}_\Gamma) = \left(\bigcup_{Q\in\mathcal{Q}_\prec} S'_Q\right) \vdash_\varepsilon^{\mathcal{T},\mathcal{V}_S} \sigma' Z'.\Gamma$.
G2. $\mathbb{T}'_\Gamma$ is (Γ, Φ)-conformant.

As $\mathbf{T}_\Gamma, \mathcal{T}$ and $\mathcal{V}_S$ are already defined, completing the construction of $\mathbb{T}'_\Gamma$ only requires that we define ρ'_Γ and λ'_Γ, which we do so that the following invariants hold for each $\mathbf{n} \in \mathbf{N}$.

I1. If $\rho'_\Gamma(\mathbf{n})$ is defined then the sequents assigned by λ'_Γ to $\mathbf{n}$ and its children are consistent with rule application $\rho'_\Gamma(\mathbf{n})$.
I2. $fm(\lambda'_\Gamma(\mathbf{n})) = fm(\mathbf{n})$
I3. $st(\lambda'_\Gamma(\mathbf{n})) \subseteq \bigcup_{Q\in\mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{n}))$
I4. $dl(\lambda'_\Gamma(\mathbf{n})) = dl_\Gamma(\mathbf{n})$

These invariants are suitably updated versions of the invariants appearing in the proof of Lemma 8.15.

The definitions of ρ'_Γ and λ'_Γ are given in a co-inductive fashion (i.e., "from the root down" rather than "the leaves up"). More specifically, the construction first assigns a sequent to $\mathbf{r}_\Gamma$, the root of $\mathbf{T}_\Gamma$ that ensures that Property G1 holds. It immediately follows that Properties I2, I3, and I4 also hold for the root. Then for any non-leaf node $\mathbf{n}$ whose sequent satisfies I2, I3, and I4, the co-induction step defines sequents for each child of $\mathbf{n}$ so that these sequents each satisfy I2, I3, and I4 and so that Property I1 holds for $\mathbf{n}$. Property G2 will be proved later.

We begin by defining $\lambda'_\Gamma(\mathbf{r}_\Gamma) = (\bigcup_{Q\in\mathcal{Q}_\prec} S'_Q) \vdash_\varepsilon^{\mathcal{T},\mathcal{V}_S} \sigma' Z'.\Gamma$. Property G1 is immediate, as is the fact that I2, I3, and I4 hold for $\mathbf{r}_\Gamma$.

For the co-inductive step, suppose $\lambda'_\Gamma(\mathbf{n}) = S_\mathbf{n} \vdash_{\Delta_\mathbf{n}}^{\mathcal{T},\mathcal{V}_S} \Gamma_\mathbf{n}$ and that invariants I2, I3, and I4 all hold for $\lambda'_\Gamma(\mathbf{n})$. We define $\lambda'_\Gamma(\mathbf{n}')$ for each child $\mathbf{n}'$ of $\mathbf{n}$, and $\rho'_\Gamma(\mathbf{n})$, the rule application for $\mathbf{n}$, so as to ensure that invariant I1 holds for $\mathbf{n}$ and that I2, I3, and I4 are established for each child $\mathbf{n}'$. The constructions in many cases closely match those found in Lemma 8.15.

***$rn_\Gamma(\mathbf{n})\bot$*, or *$rn_\Gamma(\mathbf{n}) \in \{\wedge, \vee, [K], \langle K\rangle\}$*.** The constructions in this case mirror those in Lemma 8.15 for λ_S; the only difference is that the definition lists in the child sequents are inherited from the parent, rather than always being ε. It is straightforward to see that invariant I1 holds for $\mathbf{n}$ while I2, I3, and I4 hold for the children of $\mathbf{n}$.

***$rn_\Gamma(\mathbf{n}) = \sigma Z''$*.** In this case, $fm_\Gamma(\mathbf{n}) = \sigma'' Z''.\Gamma'$ for some σ'', Z'' and Γ'; $cs(\mathbf{n}) = \mathbf{n}'$; $dl_\Gamma(\mathbf{n}') = \Delta_{\mathbf{n}'} = \Delta_\mathbf{n} \cdot (U' = fm_\Gamma(\mathbf{n}))$ for some $U' \notin \mathrm{dom}(\Delta_\mathbf{n})$; and $fm_\Gamma(\mathbf{n}') = U'$. Define $\rho'_\Gamma(\mathbf{n}) = \sigma Z''$ and $\lambda'_\Gamma(\mathbf{n}') = S_\mathbf{n} \vdash_{\Delta_{\mathbf{n}'}}^{\mathcal{T},\mathcal{V}_S} U'$. It is easy to establish that invariant I1 holds for $\mathbf{n}$ and that I2, I3, and I4 hold for $\mathbf{n}'$.

***$rn_\Gamma(\mathbf{n})$* = Un.** In this case, $cs(\mathbf{n}) = \mathbf{n}'$, $dl_\Gamma(\mathbf{n}') = \Delta_\mathbf{n}$ and $fm_\Gamma(\mathbf{n}) = U'$ for some $U' \in \mathrm{dom}(\Delta_\mathbf{n})$. Let $\Delta_\mathbf{n}(U') = \sigma'' Z''.\Gamma'$; then $fm_\Gamma(\mathbf{n}') = \Gamma'[Z'' := U']$. Define $\rho'_\Gamma(\mathbf{n}) = \mathrm{Un}$ and $\lambda'_\Gamma(\mathbf{n}') = S_\mathbf{n} \vdash_{\Delta_\mathbf{n}}^{\mathcal{T},\mathcal{V}_S} fm_\Gamma(\mathbf{n}')$. It is easy to establish that invariant I1 holds for $\mathbf{n}$ and that I2, I3, and I4 hold for $\mathbf{n}'$.

rn$_\Gamma(\mathbf{n})$ = Thin. In this case, $cs(\mathbf{n}) = \mathbf{n}'$, $dl_\Gamma(\mathbf{n}') = \Delta_\mathbf{n}$ and $fm_\Gamma(\mathbf{n}) = fm_\Gamma(\mathbf{n}')$. Take $S_{\mathbf{n}'}$ to be $\bigcup_{Q \in \mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{n}'))$, and define $\rho'_\Gamma(\mathbf{n})$ = Thin and $\lambda'_\Gamma(\mathbf{n}') = S_{\mathbf{n}'} \vdash^{\mathcal{T},\mathcal{V}_S}_{\Delta_\mathbf{n}} fm_\Gamma(\mathbf{n}')$. Invariant I1 can be shown to hold for $\mathbf{n}$, while I2, I3, and I4 hold for $\mathbf{n}'$. (Indeed, a stronger version of I3 holds in this case, as $st(\lambda'_\Gamma(\mathbf{n}')) = \bigcup_{Q \in \mathcal{Q}_\prec} st_{\Gamma,Q}(\mathbf{n}')$.)

We now argue that $\mathbb{T}'_\Gamma$ is a successful TNF tableau that is compliant with $\prec'$ and (Γ, Φ)-conformant (i.e., satisfies Property G2). We first must establish that $\mathbb{T}'_\Gamma$ is indeed a tableau. Because of invariant I1 it suffices to show that the sequent labeling any leaf node is terminal (cf. Definition 6.9(2)). Let $\mathbf{n}$ be a leaf in $\mathbf{T}_\Gamma$; we note that $fm(\mathbf{n})$ has form either Z'' or $\neg Z''$, where Z'' is free in $\sigma' Z'.\Gamma$, or U' for some definitional constant $U' \in \mathrm{dom}(dl(\mathbf{n}))$. In the first two cases $\mathbf{n}$ is clearly terminal. In the latter case we must argue that $st(\lambda'_\Gamma(\mathbf{n})) \subseteq st(\lambda'_\Gamma(\mathbf{m}))$, where $\mathbf{m}$ is the intended companion node of $\mathbf{n}$ (i.e., the strict ancestor of $\mathbf{n}$ such that $fm_\Gamma(\mathbf{m}) = fm_\Gamma(\mathbf{n}) = U'$). From the definition of $\mathbb{T}'_\Gamma$ and the fact that each $\mathbb{T}_{\Gamma,Q}$ is a TNF tableau we observe the following:

(1) $\rho'_\Gamma(\mathbf{m})$ = Un, and $\mathbf{m}$ is the only internal node whose formula is U'.
(2) Since $p_\Gamma(\mathbf{m})$ is the parent of $\mathbf{m}$, $\rho'_\Gamma(p_\Gamma(\mathbf{m})) = \sigma Z''$. for some Z''.
(3) Either $p_\Gamma(\mathbf{m})$ is the root of $\mathbf{T}_\Gamma$ (i.e., $p(\mathbf{m}) = \mathbf{r}_\Gamma$), or $p(p(\mathbf{m}))$, the grandparent of $\mathbf{m}$, is defined, and $\rho'_\Gamma(p(p(\mathbf{m})))$ = Thin. In either case, from the definition of the construction it can be shown that $st(\lambda_\Gamma(\mathbf{m})) = \bigcup_{Q \in \mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{m}))$.

Because each $\mathbb{T}_{\Gamma,Q}$ is a tableau it follows that for each $Q \in \mathcal{Q}_\prec$, $\mathbf{n}$ is terminal in $\mathbb{T}_{\Gamma,Q}$ and thus $st(\lambda_{\Gamma,Q}(\mathbf{n})) \subseteq st(\lambda_{\Gamma,Q}(\mathbf{m}))$. Also, since invariant I3 holds of $\mathbf{n}$ in $\mathbb{T}'_\Gamma$ we have that $st(\lambda'_\Gamma(\mathbf{n})) \subseteq \bigcup_{Q \in \mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{n}))$. We can now reason as follows:

$$st(\lambda'_\Gamma(\mathbf{n})) \subseteq \bigcup_{Q \in \mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{n})) \subseteq \bigcup_{Q \in \mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{m})) = st(\lambda'_\Gamma(\mathbf{m}))$$

to see that $\mathbf{n}$ is terminal in $\mathbb{T}'_\Gamma$ and thus $\mathbb{T}'_\Gamma$ is indeed a tableau.

We now show that $\mathbb{T}'_\Gamma$ is successful, compliant with $\prec'$, and (Γ, Φ)-conformant. Before doing that, however, we remark on properties of dependency relations in $\mathbb{T}'_\Gamma$ that will be used in the arguments to follow. To begin with, the following holds of all $\mathbf{n}_1, \mathbf{n}_2, s_1$, and s_2 such that $s_2 <_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}'_\Gamma$:

for every $Q \in \mathcal{Q}_\prec$ such that $s_1 \in st(\lambda_{\Gamma,Q}(\mathbf{n}_1))$, either $s_2 \in st(\lambda_{\Gamma,Q}(\mathbf{n}_2))$, and thus, $s_2 <_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}_{\Gamma,Q}$, or there exists Q' such that $Q' \sqsubset Q$ such that $s_2 \in st(\lambda_{\Gamma,Q'}(\mathbf{n}_2))$.

This is immediate from the definition of $<_{\mathbf{n}_2,\mathbf{n}_1}$ (Definition 6.14) and $\mathbb{T}'_\Gamma$: the need for the Q' case comes from the construction used for rules $\vee$ and $\langle K \rangle$.[10] An inductive argument based on the definition of $<:$ lifts this result to $\mathbf{n}_1, \mathbf{n}_2, s_1$, and s_2 such that $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_2$ in $\mathbb{T}'_\Gamma$: for all $Q \in \mathcal{Q}_\prec$ such that $s_1 \in st(\lambda_{\Gamma,Q}(\mathbf{n}_1))$, either $s_2 \in st(\lambda_{\Gamma,Q}(\mathbf{n}_2))$ and $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_1$ in $\mathbb{T}_{\Gamma,Q}$, or there exists $Q' \sqsubset Q$ such that $s_2 \in st(\lambda_{\Gamma,Q'}(\mathbf{n}_2))$.

We now establish that $\mathbb{T}'_\Gamma$ is successful by showing that every leaf in $\mathbb{T}'_\Gamma$ is successful (cf. Definition 6.13), which amounts to showing that for each leaf $\mathbf{n}$, $st(\lambda'_\Gamma(\mathbf{n})) \subseteq \| fm_\Gamma(\mathbf{n}) \|^{\mathcal{T}}_{\mathcal{V}_S}$. There are four cases to consider.

fm$_\Gamma(\mathbf{n}) = Z$. Analogous to the same case in the proof of Lemma 8.15.
fm$_\Gamma(\mathbf{n}) \in \{Z'', \neg Z''\}$ **for some** $Z'' \neq Z$ **free in** $\sigma' Z'.\Gamma$. Analogous to the same case in the proof of Lemma 8.15.
n is a ν**-leaf.** In this case, $fm_\Gamma(\mathbf{n})$ is successful by definition.
n is a μ**-leaf.** Let $\mathbf{m}$ be the companion node of $\mathbf{n}$; we must show that $<:_\mathbf{m}$ is well-founded in $\mathbb{T}'_\Gamma$. Suppose to the contrary that this is not the case, i.e., that there is an infinite descending chain

[10] An analogous property is used in the proof of Lemma 8.15.

$\cdots <:_{\mathbf{m}} s_2 <:_{\mathbf{m}} s_1$ with each $s_i \in st(\lambda'_\Gamma((\mathbf{m}))$. From the definition of $<:$ (cf. Definition 6.16) it follows that for all $j > 1$, $s_j <:_{\mathbf{n_j},\mathbf{m}} s_{j-1}$ in $\mathbb{T}'_\Gamma$ for some companion leaf $\mathbf{n}_j$ of $\mathbf{m}$ (note that $\mathbf{n}$ is one of these $\mathbf{n}_j$). Since each $\mathbb{T}_{\Gamma,Q}$ is successful we know that $<:_{\mathbf{m}}$ in $\mathbb{T}_{\Gamma,Q}$ is well-founded for any $Q \in \mathcal{Q}_\prec$. Recall also that $\sqsubset$ is a well-ordering on $\mathcal{Q}_\prec$. Now consider $\cdots <:_{\mathbf{m}} s_2 <:_{\mathbf{m}} s_1$. Since $s_1 \in st(\lambda'_\Gamma(\mathbf{m}))$ invariant I3 guarantees that $s_1 \in st(\lambda_{\Gamma,Q}(\mathbf{n}_1))$ for some $Q \in \mathcal{Q}_\prec$. Now consider the s_j, $j > 1$. Since each $s_j <:_{\mathbf{n}_j,\mathbf{m}} s_1$ in $\mathbb{T}'_\Gamma$ the preceding argument ensures that either $s_j <:_{\mathbf{n},\mathbf{m}} s_1$ in $\mathbb{T}_{\Gamma,Q}$, and thus $s_j <:_{\mathbf{m}} s_1$ in $\mathbb{T}_{\Gamma,Q}$, or there exists $s' \in S$ such that $[s'] \sqsubset Q$ and $s_j \in st(\lambda_{\Gamma,[s']}(\mathbf{n}_j))$. However, $<:_{\mathbf{m}}$ in $\mathbb{T}_{\Gamma,Q}$ is well-founded, so only finitely many of the s_j can satisfy $s_j <:_{\mathbf{m}} s_1$ in $\mathbb{T}_{\Gamma,Q}$; there must be some $j > 1$ such that $s_j \in st(\lambda_{\Gamma,[s']}(\mathbf{n}_2))$ some $[s'] \sqsubseteq Q$. But then, for $\cdots <:_{\mathbf{m}} s_2 <:_{\mathbf{m}} s_1$ to be an infinite descending chain there must be an infinite descending chain in $\sqsubset$. As $\sqsubset$ is well-founded, this is a contradiction, and $<:_{\mathbf{m}}$ must be well-founded in $\mathbb{T}'_\Gamma$, meaning $\mathbf{n}$ is a successful leaf.

To prove compliance of $\mathbb{T}'_\Gamma$ with $\prec'$, assume that $s_2 <:_{\mathbf{r}_\Gamma} s_1$ in $\mathbb{T}'_\Gamma$; we must show that $s_2 \prec' s_1$. Let $\mathbf{n}$ be a companion leaf of $\mathbf{r}_\Gamma$ such that $s_2 <:_{\mathbf{n},\mathbf{r}_\Gamma} s_1$. From the arguments above we know that $s_1 \in st(\lambda_{\Gamma,Q}(\mathbf{r}_\Gamma))$ for some $Q \in \mathcal{Q}_\prec$; assume further Q is the minimum such element in $\mathcal{Q}_\prec$ with respect to $\sqsubset$ (which is guaranteed to exist because $\sqsubset$ is a well-ordering on $\mathcal{Q}_\prec$). Also, either $s_2 <:_{\mathbf{n},\mathbf{r}_\Gamma} s_1$ in $\mathbb{T}_{\Gamma,Q}$ or there is $Q' \sqsubset Q$ such that $s_2 \in st(\lambda_{\Gamma,Q'}(\mathbf{n}))$. In the former case $s_2 <:_{\mathbf{r}_\Gamma} s_1$, and the fact that $\mathbb{T}_{\Gamma,Q}$ is successful and compliant with $\prec'_Q \subseteq \prec'$ guarantees that $s_2 \prec' s_1$. In the latter case, since $\mathbf{n}$ is a companion leaf of $\mathbf{r}_\Gamma$ we know that $s_2 \in st(\lambda_{\Gamma,Q}(\mathbf{r}_\Gamma))$ and $s_2 \in st(\lambda_{\Gamma,Q'}(\mathbf{r}_\Gamma))$. Moreover, the fact that Q is minimum ensures that $s_1 \notin st(\lambda_{\Gamma,Q'}) = S'_{Q'} = ||\,\sigma' Z'.\Gamma\,||^{\mathcal{T}}_{\mathcal{V}_{Q'}}$. However, since $\prec'$ is locally consistent with $\prec$ means that $s_1 \not\prec' s_2$, and as $\prec'$ is total it must be that $s_2 \prec' s_1$.

We now prove (Γ,Φ)-conformance for $\mathbb{T}'_\Gamma$. So fix $x \in S$, $y \in S'_{[x]}$ and $x' <:_{\mathbf{n}',\mathbf{r}_\Gamma} y$ where $fm_\Gamma(\mathbf{n}') = Z$. Note that $x' \in \prec^{-1}(S)$. We must show that $x' \prec x$. From facts established above we know that either $x' <:_{\mathbf{n}',\mathbf{r}_\Gamma} y$ in $\mathbb{T}_{\Gamma,[x]}$, or there exists a $Q \sqsubset [x]$ such that $x' \in st(\lambda_{\Gamma,Q}(\mathbf{n}'))$. In the former case, the success of $\mathbb{T}'_{\Gamma,Q}$ guarantees that $x' \in \mathcal{V}_{[x]}(Z) = \prec^{-1}([x])$, meaning $x' \prec x$. In the latter case, $x' \in \mathcal{V}_Q(Z) = \prec^{-1}(Q)$; since $\prec$ is total and $Q \sqsubset [x]$ it follows that $\prec^{-1}(Q) \subseteq \prec^{-1}([x])$, so $x' \in \prec^{-1}([x])$ and $x' \prec x$.

Construction of $\mathbb{T}_\Gamma$. We now show how to construct successful TNF tableau $\mathbb{T}_\Gamma$ for sequent $S' \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}_S} \sigma' Z'.\Gamma$ that is compliant with $\prec'$ and (Γ,Φ)-conformant. We begin by noting that since $(S',\prec')$ is a σ'-compatible, total qwf support structure for g_S, the induction hypothesis and Lemma 8.12 guarantee the existence of a successful TNF tableau

$$\mathbb{T}_{\Gamma,S} = (\mathbf{T}, \rho_{\Gamma,S}, \mathcal{T}, \mathcal{V}_S, \lambda_{\Gamma,S})$$

for sequent $S' \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}_S} \sigma' Z'.\Gamma$ that is compliant with $\prec'$ and structurally equivalent to $\mathbb{T}_{\Gamma,Q}$ for any $Q \in \mathcal{Q}_\prec$. We now build $\mathbb{T}_\Gamma$ using a coinductive definition of λ_Γ that merges $\mathbb{T}'_\Gamma$ and $\mathbb{T}_{\Gamma,S}$ in a node-by-node fashion, starting with $\mathbf{r}_\Gamma$, the root, so that the resulting tableau is (Γ,Φ)-conformant. The construction also ensures that the invariants below, which are adapted from the construction of $\mathbb{T}'_\Gamma$, also hold.

I1. If $\rho_\Gamma(\mathbf{n})$ is defined then the sequents assigned by λ_Γ to $\mathbf{n}$ and its children are consistent with the rule application $\rho_\Gamma(\mathbf{n})$.
I2. $fm(\lambda_\Gamma(\mathbf{n})) = fm(\mathbf{n})$.
I3. $st(\lambda_\Gamma(\mathbf{n})) \subseteq st(\lambda'_\Gamma(\mathbf{n})) \cup st(\lambda_{\Gamma,S}(\mathbf{n}))$.
I4. $dl(\lambda_\Gamma(\mathbf{n})) = dl_\Gamma(\mathbf{n})$.

(That is, λ'_Γ is replaced by λ_Γ, and $\bigcup_{Q\in\mathcal{Q}_\prec} st(\lambda_{\Gamma,Q}(\mathbf{n}))$ is replaced by $st(\lambda'_\Gamma(\mathbf{n})) \cup st(\lambda_{\Gamma,S}(\mathbf{n}))$.) As before, the definition of λ_Γ begins by assigning a value to $\lambda_\Gamma(\mathbf{r}_\Gamma)$ so that invariants I2–I4 are satisfied. The coinductive step then assumes that $\mathbf{n}$ satisfies these invariants and defines λ_Γ for the children of $\mathbf{n}$ and $\rho_\Gamma(\mathbf{n})$ so that I1 holds for $\mathbf{n}$ and I2–I4 hold for each child. The details of the construction are very similar to those for $\mathbb{T}'_\Gamma$ and are omitted, as are the arguments that $\mathbb{T}_\Gamma$ is successful, compliant with $\prec'$, and (Γ, Φ)-conformant.

Step 4 of proof outline: construct tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$. To complete the proof we construct tableau $\mathbb{T}_\Phi = (\mathbf{T}_\Phi, \rho_\Phi, \mathcal{T}, \mathcal{V}, \lambda_\Phi)$, where $\mathbf{T}_\Phi = (\mathbf{N}_\Phi, \mathbf{r}_\Phi, p_\Phi, cs_\Phi)$, from $\mathbb{T}_{\Phi'}$ and $\mathbb{T}_\Gamma$ and establish that it is successful and compliant with $(S, \prec)$. Without loss of generality we assume that $\mathbf{N}'_\Phi \cap \mathbf{N}_\Gamma = \emptyset$, and define $\mathbf{N}_\Phi = \mathbf{N}_{\Phi'} \cup \mathbf{N}_\Gamma$. The construction of $\mathbb{T}_\Phi$ essentially works by embedding $\mathbb{T}_\Gamma$ as a sub-tableau of $\mathbb{T}_{\Phi'}$ underneath the leaf $\mathbf{n}_W$ of $\mathbb{T}'_\Phi$ whose formula is W. Some additional housekeeping details are necessary to ensure the result is a tableau in TNF.

— Suppose $\mathbf{n} \in N_{\Phi'}$ and $\lambda_{\Phi'}(\mathbf{n}) = X \vdash^{\mathcal{T},\mathcal{V}'}_{\Delta} \Phi'$. Then $\lambda_\Phi(\mathbf{n}) = X \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi'[W := \sigma' Z'.\Gamma]$. This replaces all instances of W by $\sigma' Z'.\Gamma$ and changes the valuation from $\mathcal{V}'$ to $\mathcal{V}$.
— Set $\rho_\Phi(\mathbf{n}_W) = \text{Thin}$; for all other nodes in $\mathbf{N}_\Phi$ use the value of $\rho_{\Phi'}$ or ρ_Γ as appropriate.
— Suppose $\mathbf{n} \in \mathbf{N}_\Gamma$ and $\lambda_\Gamma(\mathbf{n}) = X \vdash^{\mathcal{T},\mathcal{V}_S}_{\Delta} \Gamma'$. Then $\lambda_\Phi(\mathbf{n}) = X \vdash^{\mathcal{T},\mathcal{V}}_{(U=\sigma Z.\Phi)\cdot\Delta} \Gamma'[Z := U]$. This adjusts the definition list in $\mathbf{n}$, replaces all free occurrences of Z by U, and updates $\mathcal{V}_S$ to $\mathcal{V}$.

To complete the proof of the lemma we must show that $\mathbb{T}_\Phi$ is a successful TNF tableau that is compliant with $(S, \prec)$. That $\mathbb{T}_\Phi$ is indeed a tableau is immediate from its construction and the fact that $\mathbb{T}_{\Phi'}$ and $\mathbb{T}_\Gamma$ are tableaux: in particular, every leaf in $\mathbb{T}_\Phi$ corresponds either to a leaf in $\mathbb{T}_{\Phi'}$ or to a leaf in $\mathbb{T}_\Gamma$ and is therefore guaranteed to be terminal. The TNF property follows in a similar fashion. We now argue that $\mathbb{T}_\Phi$ is successful and compliant with $(S, \prec)$. We begin by noting that since $\mathbb{T}_{\Phi'}$ and $\mathbb{T}_\Gamma$ are successful, every leaf $\mathbf{n}$ such that $fm(\lambda_\Phi(\mathbf{n})) \neq U$, where U is the definitional constant associated with $\sigma Z.\Phi$, is also successful in $\mathbb{T}_\Phi$. Proving that $\mathbb{T}_\Phi$ is successful therefore reduces to proving the success of each U-leaf. If we can show compliance of $\mathbb{T}_\Phi$ with $(S, \prec)$, then the success of each U-leaf also follows, for in the particular case when $\sigma = \mu$, the well-foundedness of $<:_{\mathbf{r}'}$, where $\mathbf{r}' = cs(\mathbf{r}_\Phi)$ is the unique (due to TNF) U-companion node in $\mathbb{T}_\Phi$, follows immediately from the well-foundedness of $\prec$. To this end, suppose that $s, s' \in S$ are such that $s' <:_{\mathbf{r}'} s$; we must show that $s' \prec s$. Since $s' <: \mathbf{r}' s$ holds there must exist a U-leaf $\mathbf{n}$ such that $s' \in st(\lambda_\Phi(\mathbf{n}))$ and $s' <:_{\mathbf{n},\mathbf{r}'} s$ in $\mathbb{T}_\Phi$. There are two cases to consider.

(1) If $\mathbf{n}'$ is a leaf in $\mathbb{T}_{\Phi'}$, the compliance of $\mathbb{T}_{\Phi'}$ with respect to $(S, \prec)$ guarantees that $s' \prec s$.
(2) If $\mathbf{n}'$ is a leaf in $\mathbb{T}_\Gamma$, there must exist $y \in \|\, \sigma' Z'.\Gamma \,\|^{\mathcal{T}}_{\mathcal{V}[s]}$ such that $s' <:_{\mathbf{n},\mathbf{r}_\Gamma} y$ in $\mathbb{T}_\Phi$, and thus, $\mathbb{T}_\Gamma$, and $y <:_{\mathbf{r}_\Gamma,\mathbf{r}'} s$. Since $\mathbb{T}_\Gamma$ is (Γ, Φ)-conformant it must follow in this case that $s' \prec s$.

This completes the proof. □

With these lemmas in hand we may now state and prove the completeness theorem.

Theorem 8.18 (Completeness). *Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be an LTS and $\mathcal{V}$ a valuation, and let S and Φ be such that $S \subseteq \|\, \Phi \,\|^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi$ has a successful tableau.*

Proof. The proof proceeds as follows. Let $\sigma_1 Z_1.\Phi_1, \ldots, \sigma_n Z_n.\Phi_n$ be the top-level fixpoint subformulas of Φ, and let $W_1, \ldots, W_n$ be fresh variables. Define Φ' to be the fixpoint-free formula containing exactly one occurrence of each W_i and such that

$$\Phi = \Phi'[W_1, \ldots, W_n := \sigma_1 Z_1.\Phi_1, \ldots \sigma_n Z_n.\Phi_2].$$

Also define $S_i = \|\, \sigma_i Z_i.\Phi_i \,\|^{\mathcal{T}}_{\mathcal{V}}$ for each $i = 1, \ldots n$, and let $\mathcal{V}'$ be defined by

$$\mathcal{V}' = \mathcal{V}[W_1, \ldots W_n := S_1, \ldots S_N].$$

Lemma 8.17 guarantees a successful tableau $\mathbb{T}'$ for $S \vdash^{\mathcal{T},\mathcal{V}'}_{\varepsilon} \Phi'$.

Also, for each $i = 1, \ldots, n$ define $(S_i, \prec_i)$ to be a σ_i-maximal support structure for $\| Z_i.\Phi_i \|_{\mathcal{V}}^{\mathcal{T}}$. Lemma 8.17 guarantees a successful tableau $\mathbb{T}_i$ for each sequent $S_i \vdash_{\varepsilon}^{\mathcal{T},\mathcal{V}} \sigma_i Z_i.\Phi_i$. We may now construct a successful tableau $\mathbb{T}$ by making each leaf in $\mathbb{T}'$ whose formula is W_i the parent of the root of tableau $\mathbb{T}_i$, setting the rule for this former leaf node to be Thin and replacing all occurrences of W_i by $\sigma_i Z_i.\Phi_i$. The resulting tableau is guaranteed to be successful. □

9 PROOF SEARCH

The previous sections have given an alternative formalization of a proof system for the modal mu-calculus and proven it sound and complete. In this section and the next, we show how our approach can be adapted in various useful ways. This section focuses on an alternative success condition to the general one in Section 6 that enables proof search to be automated in some cases.

From an application perspective, the most difficult aspect of the proof system in this article is the need to check the well-foundedness of the extended-dependency orderings for nodes labeled with least fixed-point formulas. In the abstract setting of infinite-state LTSs considered in this article, unfortunately there is really no alternative. In many situations, however, this well-foundedness can be manifest from additional semantic information present in the definition of the transition relations. One interesting such case involves modifying the success criterion for μ-leaves so that they are only successful if their set of states is empty. Intuitively, this means that during proof construction we simply keep unfolding nodes labeled with definitional constants that are associated with a least fixpoint formula until we reach a node with an empty state set. Of course, with this modification, the unfolding procedure does not necessarily terminate, and hence the proof system is not complete for infinite-state LTSs in general. However, for instance for the class of infinite-state transition systems induced by timed automata, the resulting tableaux method is complete [67] for the alternation-free mu-calculus. Furthermore, even in case the modification is not complete, the result is a sound semi-decision procedure in the sense that, if a (finite) proof tree is obtained, the proof tree is successful.

We first strengthen the notion of successful tableau (Definition 6.13) in such a way that a μ-leaf is only successful if it has an empty set of states.

Definition 9.1 (Strongly Successful Complete Tableau). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a complete tableau. Then leaf node $\mathbf{n}$ in $\mathbb{T}$ is *strongly successful* iff one of the following holds.

(1) $fm(\mathbf{n}) = Z$ for some $Z \in \text{dom}(dl(\mathbf{n}))$ and $st(\mathbf{n}) \subseteq \mathcal{V}(Z)$.
(2) $fm(\mathbf{n}) = \neg Z$ for some $Z \in \text{dom}(dl(\mathbf{n}))$ and $st(\mathbf{n}) \cap \mathcal{V}(Z) = \emptyset$.
(3) $\mathbf{n}$ is a ν-leaf; or
(4) $\mathbf{n}$ is a μ-leaf with $st(\mathbf{n}) = \emptyset$.

$\mathbb{T}$ is strongly successful iff all its leaves are strongly successful.

The only difference between strongly successful tableaux and successful tableaux (Definition 6.13) is in the μ-leaves. For the μ-leaves, our new criterion gives a much simpler formulation, that is, the set of states in a strongly successful μ-leaf is empty. As every μ-node $\mathbf{n}$ in a strongly successful complete tableau has an empty state set, it can be seen that relation $<:_{\mathbf{m}}$ for μ-companion node $\mathbf{m}$ is $\emptyset$ and, therefore, trivially well-founded. Thus, every strongly successful complete tableau is also successful. We, therefore, have the following:

Theorem 9.2. *Fix LTS $(\mathcal{S}, \rightarrow)$ of sort Σ and valuation $\mathcal{V}$. Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a strongly successful complete tableau for sequent $\mathbf{s} \in \mathbf{S}_{\text{Var}}^{\mathcal{T}}$, where $dl(\mathbf{s}) = \varepsilon$. Then $\mathbf{s}$ is valid.*

Proof. Immediate from Theorem 7.20 and the fact that every strongly successful complete tableau is also a successful tableau. □

As noted above the proof system with the modified success criterion given in this section is not in general complete, although cases where it is complete can be characterized [25, 66].

We close this section by noting that our ability to alter the definition of successful tableau to successful strongly complete tableau is due to our formalization of trees and tableaux. In particular, our proofs permit the application of proof rules to nodes even if the nodes are terminal. In the setting of [13], it is not clear if this use of proof rules is allowed.

10 A TIMED MODAL MU-CALCULUS

This section gives another illustration of the utility of the mu-calculus proof-system framework developed in this article. In particular, we show how to extend the proof system described in Section 6 to *TTSs* and properties expressed in a timed extension of the mu-calculus, which enriches the mu-calculus with two modalities for expressing properties of continuous timed behavior. Because of their structure, the proofs of soundness and completeness given earlier only need to be extended with cases for the new modal operators to cover the timed setting.

We begin by defining TTSs and the extension of the mu-calculus to be considered. TTSs are infinite-state transition systems used to give semantics to, for example, timed and hybrid automata. Besides introducing transitions labeled by time, TTSs also capture continuous as well as discrete system behavior.

Definition 10.1 (Timed Sort). Sort Σ is a *timed sort* iff $\mathbb{R}_{\geq 0} \subseteq \Sigma$, i.e., every non-negative real number is an element of Σ. If Σ is a timed sort we write $act(\Sigma) = \Sigma \setminus \mathbb{R}_{\geq 0}$ for the set of non-numeric elements in Σ. We sometimes refer to elements of $act(\Sigma)$ as *actions* and $\delta \in \mathbb{R}_{\geq 0}$ as *time delays.*

Definition 10.2 (TTS [9]). LTS $(\mathcal{S}, \rightarrow)$ of timed sort Σ is a *TTS* if it satisfies the following conditions:

(1) For all states $s \in \mathcal{S}$, $s \xrightarrow{0} s$, i.e., the transition system is *time-reflexive.*
(2) For all states $s, s', s'' \in \mathcal{S}$ and $\delta \in \mathbb{R}_{\geq 0}$ if $s \xrightarrow{\delta} s'$ and $s \xrightarrow{\delta} s''$ then $s' = s''$, i.e., the transition system is *time-deterministic.*
(3) For all states $s, s', s'' \in \mathcal{S}$ and $\delta, \delta' \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\delta} s'$ and $s' \xrightarrow{\delta'} s''$ then $s \xrightarrow{\delta+\delta'} s''$, i.e., the transition system is *time-additive.*
(4) For all states $s, s' \in \mathcal{S}$ and $\delta \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\delta} s'$ then for all $\delta', \delta'' \in \mathbb{R}_{\geq 0}$ with $\delta = \delta' + \delta''$, there exists $s'' \in \mathcal{S}$ such that $s \xrightarrow{\delta'} s''$ and $s'' \xrightarrow{\delta''} s'$, i.e., the transition system is *time-continuous.*

In a TTS a transition can either be labeled by an action, i.e., those labels in $act(\Sigma)$, or a time delay $\delta \geq 0$. Intuitively, if a system is in state s and $s \xrightarrow{\delta} s'$ then the system is in state s' after δ units of time have elapsed, assuming no action has occurred in the interim.

The timed mu-calculus of [25] extends the mu-calculus from Definition 5.6 with a timed modal operator $\forall_{\Phi_1}(\Phi_2)$. This operator is analogous to the release operator of LTL, in the sense that for $\forall_{\Phi_1}(\Phi_2)$ to hold of a state, either Φ_2 must hold along every time instant of the time trajectory emanating from s, or there is a point along the trajectory in which Φ_1 holds, which releases the system from having to maintain Φ_2. These intuitions are formalized below.

Definition 10.3 (Timed Mu-calculus Syntax [25]). Let Σ be a timed sort and Var a countably infinite set of propositional variables. Then formulas of the timed modal mu-calculus over Σ and Var are given by the following grammar:

$$\Phi ::= Z \mid \neg\Phi' \mid \Phi_1 \wedge \Phi_2 \mid [K]\Phi' \mid \forall_{\Phi_1}(\Phi_2) \mid \nu Z.\Phi',$$

where $K \subseteq \Sigma$, $Z \in$ Var, and formulas of the form $\nu Z.\Phi'$, Z are such that Z must be positive in Φ'.

Besides the standard dualities in Section 5.2, we also have the following dual for $\forall_{\Phi_1}(\Phi_2)$:

$$\exists_{\Phi_1}(\Phi_2) = \neg\forall_{\neg\Phi_1}(\neg\Phi_2).$$

Also in analogy with LTL, $\exists_{\Phi_1}(\Phi_2)$ can be seen as an until operator: specifically, a state satisfies $\exists_{\Phi_1}(\Phi_2)$ if Φ_2 is true after some time delay from s, and until that time instant Φ_1 must be true.

To define the semantics of the timed mu-calculus, we fix some notation for time delays.

Definition 10.4 (Time Delays/successors). Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be a TTS of timed sort Σ, and let $\delta \in \mathbb{R}_{\geq 0}$.

- We define $succ \in \mathcal{S} \times \mathbb{R}_{\geq 0} \rightarrow_{\perp} \mathcal{S}$ by $succ(s, \delta) = s'$ iff $s \xrightarrow{\delta} s'$. This is well-formed because of the time-determinacy of $\rightarrow$. Note $succ(s, \delta)\perp$ iff $s \overset{\delta}{\not\rightarrow}$. Also, time continuity implies that if $\delta' \geq \delta$ and $succ(s, \delta)\perp$ then $succ(s, \delta')\perp$, and if $\delta' \leq \delta$ and $succ(s, \delta) \in \mathcal{S}$ then $succ(s, \delta') \in \mathcal{S}$.
- If $s \in \mathcal{S}$ then we write

$$\begin{aligned} del(s) &= \{\delta \in \mathbb{R}_{\geq 0} \mid s \xrightarrow{\delta}\} \\ succ(s) &= \{succ(s, \delta) \mid \delta \in del(s)\} \\ succ_{<}(s, \delta) &= \{succ(s, \delta') \mid \delta' < \delta \wedge \delta' \in del(s)\} \\ succ_{\leq}(s, \delta) &= \{succ(s, \delta') \mid \delta' \leq \delta \wedge \delta' \in del(s)\} \end{aligned}$$

  for all possible delays, the states reachable by any delay, the states reachable by delays less than δ, and states reachable by delays less than or equal to δ, respectively, from s.

For the semantics of the timed mu-calculus, we follow the definition in [26].

Definition 10.5 (Timed Mu-calculus Semantics). Let $\mathcal{T} = (\mathcal{S}, \rightarrow)$ be a TTS of sort Σ and $\mathcal{V} \in \text{Var} \rightarrow 2^{\mathcal{S}}$ a valuation. Then the semantic function $|| \Phi ||_{\mathcal{V}}^{\mathcal{T}} \subseteq \mathcal{S}$, where Φ is a timed mu-formula, is defined as in Definition 5.9, extended with the following clause:

$$|| \forall_{\Phi_1}(\Phi_2) ||_{\mathcal{V}}^{\mathcal{T}} = \{s \in \mathcal{S} \mid \forall \delta \in del(s) \colon \left(succ_{<}(s, \delta) \cap || \Phi_1 ||_{\mathcal{V}}^{\mathcal{T}}\right) = \emptyset \implies succ(s, \delta) \in || \Phi_2 ||_{\mathcal{V}}^{\mathcal{T}}\}.$$

Intuitively, s satisfies $\forall_{\Phi_1}(\Phi_2)$ if for every possible delay transition from s, either the target state satisfies Φ_2, or there is a delay transition of smaller duration whose target state satisfies Φ_1, thereby releasing s of the responsibility of keeping Φ_2 true beyond that delay. For the dual operator one may derive the following semantic equivalence:

$$|| \exists_{\Phi_1}(\Phi_2) ||_{\mathcal{V}}^{\mathcal{T}} = \{s \in \mathcal{S} \mid \exists \delta \in del(s) \colon succ_{<}(s, \delta) \subseteq || \Phi_1 ||_{\mathcal{V}}^{\mathcal{T}} \wedge succ(s, \delta) \in || \Phi_2 ||_{\mathcal{V}}^{\mathcal{T}}\}.$$

Based on this characterization, one can see that $\exists_{\Phi_1}(\Phi_2)$ captures a notion of until. Specifically, s satisfies $\exists_{\Phi_1}(\Phi_2)$ if there is a delay transition from s leading to a state satisfying Φ_2, and all delay transitions of strictly shorter duration from s lead to states satisfying Φ_1.

We now extend our proof system to the timed mu-calculus setting. The notions of *timed definition list* and *timed sequent* are the obvious generalizations of the notions in Definitions 6.1 and 6.6, as is the notion of the semantics $|| \mathbf{s} ||$ of timed sequent $\mathbf{s}$, which generalizes Definition 6.7. The sequent extractor functions *st*, *dl* and *fm* also carry over to the timed setting in the expected fashion.

To obtain proof rules for the timed mu-calculus, we extend those from Figure 1 with the two rules named $\forall$ and $\exists$ as shown in Figure 2. Like rule $\langle K \rangle$, rule $\exists$ uses a *witness function* f. In this case f is intended to identify, for every $s \in S$, a *witness delay* $f(s) \in del(s)$ allowed from s such that the state $succ(s, f(s))$ reached by delaying $f(s)$ time units from s satisfies Φ_2 and also such that all states reached from s using smaller delays, i.e., those states in $succ_{<}(s, f(s))$, must satisfy Φ_1.

Proof rule $\forall$ also uses a witness function g, but its functionality is a bit more complicated than the witness function used in rule $\exists$. In particular, g takes two arguments, a state and a delay, and

$$\exists \ \frac{S \vdash_\Delta \exists_{\Phi_1}(\Phi_2)}{f_<(S) \vdash_\Delta \Phi_1 \quad f_=(S) \vdash_\Delta \Phi_2}\ f \in S \to \mathbb{R}_{\geq 0}$$

$$\forall \ \frac{S \vdash_\Delta \forall_{\Phi_1}(\Phi_2)}{g_<(S) \vdash_\Delta \Phi_1 \quad g_=(S) \vdash_\Delta \Phi_2}\ g \in S \times \mathbb{R}_{\geq 0} \to_\perp \mathbb{R}_{\geq 0}$$

where

- f is such that for all $s \in S$, $f(s) \in \mathit{del}(s)$, and $f_<(S), f_=(S) \subseteq \mathcal{S}$ are defined as
$$f_<(S) = \bigcup_{s \in S} \mathit{succ}_<(s, f(s))$$
$$f_=(S) = \{\mathit{succ}(s, f(s)) \mid s \in S\}.$$
- g is such that for all $s \in S$ and $\delta \in \mathit{del}(s)$, $g(s,\delta) \leq \delta$, and $g_<(S), g_=(S) \subseteq \mathcal{S}$ are defined as
$$g_<(S) = \{\mathit{succ}(s, g(s,\delta)) \mid s \in S \wedge \delta \in \mathit{del}(s) \wedge g(s,\delta) < \delta\}$$
$$g_=(S) = \{\mathit{succ}(s, g(s,\delta)) \mid s \in S \wedge \delta \in \mathit{del}(s) \wedge g(s,\delta) = \delta\}$$

Fig. 2. Proof rules for timed modal operators (extends Figure 1).

while g may be partial, it is required that for any $s \in S$, $g(s,\delta)$ is defined for every $\delta \in \mathit{del}(s)$. If $\delta \in \mathit{del}(s)$ and $g(s,\delta) < \delta$ then $g(s,\delta)$ is intended to be a delay such that the state reached from s via that delay satisfies the "release" formula Φ_1. If $g(s,\delta) = \delta$, then the implication is that no shorter delay exists for establishing Φ, and the state reached from s after δ has not been released from the obligation to keep Φ_2 true.

We sometimes refer to the function f mentioned in the $\exists$ rule as an $\exists$-*function* and the function g referred to in the $\forall$ rule as a $\forall$-*function*. In what follows we use

$$\mathrm{RAppl}_T = \mathrm{RAppl} \cup \{(\exists, f) \mid f \text{ is an } \exists\text{-function}\} \cup \{(\forall, g) \mid g \text{ is a } \forall\text{-function}\}$$

for the set of rule applications for the timed mu-calculus. The definitions of partial and complete tableaux (Definition 6.9) carry over immediately to partial and complete *timed* tableaux, with RAppl_T replacing RAppl appropriately. Observe that, in particular, no new terminal nodes need to be added: due to time reflexivity, $s \in \mathit{succ}(s)$ for any state s, and thus type-correct $\exists$- and $\forall$-functions can always be given when applying the $\exists$ and $\forall$ rules. (Of course the chosen functions may not necessarily lead to *successful* tableaux.) Likewise, the definitions of successful terminals and successful timed tableaux are the obvious adaptations of the notions given in Definition 6.13. If $\mathbf{n}$ is a node in a timed partial tableau then the semantics $\|\,\mathbf{n}\,\|$ of $\mathbf{n}$ carries over straightforwardly.

We now extend the local dependency ordering (Definition 6.14) to timed tableaux as follows:

Definition 10.6 (Timed Local Dependency Ordering). Let $\mathbf{n}, \mathbf{n}'$ be proof nodes in timed tableau $\mathbb{T}$, with $\mathbf{n}' \in c(\mathbf{n})$ a child of $\mathbf{n}$. Then $s' <_{\mathbf{n}',\mathbf{n}} s$ iff $s' \in \mathit{st}(\mathbf{n}')$, $s \in \mathit{st}(\mathbf{n})$, and one of the following hold:

(1) $\rho(\mathbf{n}) = [K]$ and $s \xrightarrow{K} s'$; or
(2) $\rho(\mathbf{n}) = (\langle K\rangle, f)$ and $s' = f(s)$; or
(3) $\mathit{rn}(\rho(\mathbf{n})) \notin \{[K], \langle K\rangle, \forall, \exists\}$ and $s = s'$; or
(4) $\rho(\mathbf{n}) = (\exists, f)$, $\mathit{cs}(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$, and either
  — $\mathbf{n}' = \mathbf{n}_1$ and $s' \in \mathit{succ}_<(s, f(s))$, or
  — $\mathbf{n}' = \mathbf{n}_2$ and $s' = \mathit{succ}(s, f(s))$; or
(5) $\rho(\mathbf{n}) = (\forall, g)$, $\mathit{cs}(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$, and either
  — $\mathbf{n}' = \mathbf{n}_1$ and $s' = \mathit{succ}(s, g(s,\delta))$ for some $\delta \in \mathit{del}(s)$ with $g(s,\delta) < \delta$, or
  — $\mathbf{n}' = \mathbf{n}_2$ and $s' = \mathit{succ}(s, g(s,\delta))$ for some $\delta \in \mathit{del}(s)$ with $g(s,\delta) = \delta$.

The dependency ordering $\lessdot_{\mathbf{n}',\mathbf{n}}$ and extended dependency ordering $<:_{\mathbf{n}',\mathbf{n}}$ may be adapted to timed tableaux in the obvious way, using the timed local dependency ordering as a basis. It follows using the lines of the proof of Lemma 6.20 and the definitions of the $\exists$ and $\forall$ rules that the timed local dependency ordering satisfies the semantic sufficiency property.

Lemma 10.7 (Semantic Sufficiency of Timed $\lessdot_{\mathbf{n}',\mathbf{n}}$). *Let $\mathbf{n}$ be an internal proof node in partial timed tableau $\mathbb{T}$, and $s \in st(\mathbf{n})$ such that for all s' and $\mathbf{n}'$ with $s' \lessdot_{\mathbf{n}',\mathbf{n}} s$, $s' \in \|\, \mathbf{n}' \,\|$. Then $s \in \|\, \mathbf{n} \,\|$.*

10.1 Soundness

We show how to generalize the soundness results from Section 7 to timed tableaux. We first observe that the local soundness result from Lemma 7.1 carries over to the timed setting immediately, since it solely relies on the semantic sufficiency result (Lemma 10.7) for $\lessdot_{\mathbf{n},\mathbf{n}'}$. We next show how the node formulas from Definition 7.2 generalize to nodes in a timed tableau.

Definition 10.8 (Timed Node Formulas). Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a timed tableau, with $\mathbf{C}_{\mathbb{T}}$ the companion nodes of $\mathbb{T}$. For each $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$ let $Z_{\mathbf{m}}$ be a unique fresh variable, with $\mathrm{Var}_{\mathbb{T}} = \{Z_{\mathbf{m}} \mid \mathbf{m} \in \mathbf{C}_{\mathbb{T}}\}$ the set of all such variables. Then for node $\mathbf{n} \in \mathbf{N}$ formula $P(\mathbf{n})$ is defined inductively as follows. Cases 1–10 are as in Definition 7.2. The cases for the $\exists$ and $\forall$ operators are as follows:

(11) If $\rho(\mathbf{n}) = (\exists, f)$ and $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ then $P(\mathbf{n}) = \exists_{P(\mathbf{n}_1)}(P(\mathbf{n}_2))$.
(12) If $\rho(\mathbf{n}) = (\forall, g)$ and $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ then $P(\mathbf{n}) = \forall_{P(\mathbf{n}_1)}(P(\mathbf{n}_2))$.

Valuation consistency (Definition 7.4) and the results in Lemmas 7.5, 7.6, and 7.7 and Corollary 7.8 carry over immediately to the timed setting. The definitions of support dependency ordering and influence extensions of valuations (Definitions 7.9 and 7.12) and the associated Lemmas 7.11 and 7.14 and Corollary 7.15 likewise generalize to timed tableaux in the obvious way. Using these results we can prove that $(st(\mathbf{n}), <:^{+}_{\mathbf{n}})$ is a support structure for companion nodes $\mathbf{n}$ in timed tableaux, generalizing Lemma 7.16.

Lemma 10.9. *Let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a successful timed tableau with $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ a companion node of $\mathbb{T}$ and $\mathbf{n}'$ the child of $\mathbf{n}$ in $\mathbf{T}$. Also let $S = st(\mathbf{n})$. Then $(S, <:^{+}_{\mathbf{n}})$ is a support structure for $\|\, Z_{\mathbf{n}}.P(\mathbf{n}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{n}}}$.*

Proof. Fix successful timed tableau $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ and let $\mathbf{n} \in \mathbf{C}_{\mathbb{T}}$ with $S = st(\mathbf{n})$. We prove that the following statements hold for every $\mathbf{m} \in D(\mathbf{n})$ and $s \in S$; which implies the lemma.

S1. For all x such that $x \leq:_{\mathbf{m},\mathbf{n}} s$, $x \in \|\, P(\mathbf{m}) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$.
S2. If $\mathbf{m} \in \mathbf{C}_{\mathbb{T}}$, $\mathbf{m}' = cs(\mathbf{m})$ and x satisfies $x \leq:_{\mathbf{m},\mathbf{n}} s$ then $(S_x, <:_{\mathbf{m},x})$ is a support structure for $\|\, Z_{\mathbf{m}}.P(\mathbf{m}') \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$, where $S_x = (<:^{*}_{\mathbf{m}})^{-1}(x)$ and $<:_{\mathbf{m},x} = (<:^{+}_{\mathbf{m}})\lfloor S_x$.

Analogous to the proof of Lemma 7.16, the proof proceeds by tree induction on $\mathbf{T}_{\mathbf{n}}$. For $\mathbf{m} \in D(\mathbf{n})$ and $s \in S$ we have to prove that statements S1 and S2 hold. The proof proceeds by case analysis on the form of $\rho(\mathbf{m})$; all cases are completely analogous to those in the proof of Lemma 7.16, except the proofs of statement S1 in case $\rho(\mathbf{m}) \in \{(\forall, g), (\exists, f)\}$. We here give the proof for $\rho(\mathbf{m}) = (\forall, g)$, the proof for $\rho(\mathbf{m}) = (\exists, f)$ is similar, albeit slightly less involved. In this case, $\mathbf{m} = X \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \forall_{\Phi_1}(\Phi_2)$ for some X, Φ_1 and Φ_2, $cs(\mathbf{m}) = \mathbf{m}'_1\mathbf{m}'_2$, $\mathbf{m}'_i = X_i \vdash^{\mathcal{T},\mathcal{V}}_{\Delta} \Phi_i$ for $i = 1, 2$, and $X_1 = g_{<}(X)$, $S_2 = g_{=}(X)$. The induction hypothesis ensures that S1 holds for each $t \in X_i$ and $\mathbf{m}'_i$; we must show that S1 holds for $\mathbf{m}$ and s. To this end, let $x \in X$ be such that $x \leq:_{\mathbf{m},\mathbf{n}} s$; we must show that $x \in \|\, P(\mathbf{m}) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. The result follows from the semantics of $\forall$ if we prove for all $\delta \in del(x)$ that either $succ_{<}(x, \delta) \cap \|\, P(\mathbf{m}'_1) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}} \neq \emptyset$ or $succ(x, \delta) \in \|\, P(\mathbf{m}'_2) \,\|^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. So, fix arbitrary $\delta \in del(x)$. Suppose $g(x, \delta) = \delta$, and let $x' = succ(x, g(x, \delta))$; then $x' <_{\mathbf{m}'_2,\mathbf{m}} x$. The pseudo-transitivity of $\leq\!:_{\mathbf{m},\mathbf{n}}$ guarantees that $x' \leq\!:_{\mathbf{m}'_2,\mathbf{n}} s$, and the induction hypothesis ensures that $x' \in ||\, P(\mathbf{m}'_2)\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}'_2,x'}}$. Corollary 7.15 implies that $x' \in ||\, P(\mathbf{m}'_2)\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. Next, suppose $g(x,\delta) = \delta' < \delta$, and let $x'' = succ(x, g(x, \delta'))$; then $x'' \in succ_<(x, g(x, \delta))$, and $x'' <_{\mathbf{m}'_1,\mathbf{m}} x$. The pseudo-transitivity of $\leq\!:_{\mathbf{m},\mathbf{n}}$ guarantees that $x'' \leq\!:_{\mathbf{m}'_1,\mathbf{n}} s$, and the induction hypothesis ensures that $x'' \in ||\, P(\mathbf{m}'_1)\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m}'_1,x''}}$. Corollary 7.15 guarantees that $x'' \in ||\, P(\mathbf{m}'_1)\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. As $x'' \in succ_<(x, g(x, \delta))$, $succ_<(x, g(x,\delta)) \cap ||\, P(\mathbf{m}'_1)\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}} \neq \emptyset$. Hence it follows from the semantics of $\forall$ that $x \in ||\, P(\mathbf{m})\, ||^{\mathcal{T}}_{\mathcal{V}_{\mathbf{m},x}}$. □

Corollary 7.17 again immediately generalizes to timed tableaux using the previous lemma. The following proof of soundness follows the exact same line of reasoning as the proof of Theorem 7.20.

Theorem 10.10 (Soundness of Timed Mu-calculus Proof System). *Fix TTS $(\mathcal{S}, \rightarrow)$ of timed sort Σ and valuation $\mathcal{V}$, and let $\mathbb{T} = (\mathbf{T}, \rho, \mathcal{T}, \mathcal{V}, \lambda)$ be a successful timed tableau for sequent $\mathbf{s}$, where $dl(\mathbf{s}) = \varepsilon$. Then $\mathbf{s}$ is valid.*

Observe that the results in this section only ever involve adding cases for the new operators to definitions and proofs from earlier sections that use case distinction on the operators. In all of the cases we considered, the results that need to be added for these new operators are straightforward, and follow the same line of reasoning as the other operators in the mu-calculus. This illustrates the extensibility of the proof methodology we have developed, at least for proving soundness of tableaux when adding new operators to the mu-calculus.

10.2 Completeness

We now turn our attention to establishing completeness of the tableaux construction for the timed mu-calculus. The construction used to establish completeness in Section 8 can be straightforwardly adapted to account for the new modalities introduced by the timed mu-calculus. This, again, illustrates the extensibility of the proofs given in this article. Given the similarities of the proofs, and due to page limitations, we include the proofs in the appendix.

We first note that the notion of TNF introduced in Definition 8.10 carries over to timed tableaux as well; the proof of the corresponding Lemma 8.12 only needs to be adapted by including $\forall$ and $\exists$ in case that $rn(\rho(\mathbf{n})) \in \{\wedge, \vee, [K], \langle K\rangle\}$. The details of that adaptation are routine and left to the reader. The notion of tableau compliance (Definition 8.14) also generalizes to the timed setting in the obvious way. We next establish the existence of successful TNF timed tableaux for valid sequents. This lemma mirrors the analogous result (Lemma 8.17) proved earlier for the non-timed mu-calculus, and the proof is a straightforward extension of the earlier proof.

Lemma 10.11 (Timed Fixpoint Completeness). *Fix TTS $\mathcal{T}$ and timed mu-calculus formula Φ, and let $Z, \mathcal{V}, \sigma$, and S be such that $S = ||\, \sigma Z.\Phi\, ||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $||\, Z.\Phi\, ||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful TNF timed tableau compliant with $(S, \prec)$.*

Proof. This proof is completely analogous to the proof of Lemma 8.17. In those places where the proof involves merging tableaux, the construction for merging nodes with rule names $\forall$ and $\exists$ is similar to that of $\langle K\rangle$.[11] □

With these lemmas in hand we may now state and prove the completeness theorem.

[11]The appendix includes a completeness proof of formulas with a single fixed point that shows in detail for nodes with rule names $\forall$ and $\exists$ are merged.

Theorem 10.12 (Completeness). *Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be a TTS and $\mathcal{V}$ a valuation, and let $S \subseteq \mathcal{S}$ and Φ be a timed mu-calculus formula such that $S \subseteq || \Phi ||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi$ has a successful tableau.*

Proof. Analogous to the proof of Theorem 8.18, but using Lemma 10.11 instead of Lemma 8.17. □

11 CONCLUSIONS AND FUTURE WORK

The work in this article was motivated by a desire to give a sound and complete proof system for the timed mu-calculus for infinite-state systems. We intended to do so by modifying an existing proof system for infinite-state systems and the untimed mu-calculus due to Bradfield and Stirling [11, 13], but this proved difficult because of the delicacy of their soundness and completeness arguments. Instead, we have given an alternative approach, based on explicit tableau constructions, for the untimed mu-calculus. We then showed how these constructions admitted reasoning about modifications to the core proof system, including new proof-search strategies and new logical modalities, including those in the timed mu-calculus. In the end we achieved our initial goal of a sound and complete proof system for the timed mu-calculus that is built on a new sound and complete proof system for the untimed mu-calculus.

Our proof techniques are based on lattice-theoretic results that give new characterizations of fixpoints of monotonic functions over complete lattices in terms of a notion we call *support structures.* Using this approach, we are able to present proofs that, in contrast to the results in [11], do not require reasoning about ordinal unfoldings of formulas, and that is extensible to other termination conditions and modalities. Our completeness results rely on direct constructions of proof tableaux for valid sequents; this also facilitates extensibility.

We have illustrated our new approach by showing that its soundness proof straightforwardly carries over to the proof system where μ-nodes with non-empty sets of states are always unfolded. Although we have not considered this in this article, we expect this proof system can be shown to be complete for restricted but useful infinite-state systems, such as those with finite bisimulation quotients. Additionally, we have presented a proof system for an extension of the mu-calculus with two timed modalities, and shown our soundness and completeness proof immediately carry over.

Future work. The proof techniques presented in this article enable us to prove soundness of extensions and modifications of the proof system for infinite-state systems. In particular, soundness of the proof system for the timed mu-calculus opens opportunities for model checking such systems. We plan to implement this proof system, dealing with sets of states symbolically, and thus extending model checkers for alternation-free timed mu-calculi [25] for timed automata, a class of TTSs, to the full mu-calculus.

Furthermore, the proof of soundness and completeness for the timed mu-calculus only involves adding cases for the newly added operators to the proofs of results about the base proof system. Each of these cases follows the same line of reasoning as the results for other operators. We, therefore, expect that the soundness and completeness results can be generalized to only refer to properties about the local dependencies of the operators in the mu-calculus, and our support structure results. It would be interesting to investigate this direction, and simplify the proof obligations for soundness and completeness when adding operators to the mu-calculus even further.

Another direction to be explored is how the soundness and completeness results in this article can be adapted to the equational mu-calculus [20], and other equational theories such as Boolean equation systems [50] and parameterized Boolean equation systems [29, 30]. In these formalisms there are already unique identifiers (the left-hand side variables of the equations) for each fixpoint in the formalism, thereby possibly obviating the need for definition lists in the proof systems. The mechanism of unfolding in the proof system needs to be adapted to deal with this difference.

Finally, another common extension of the mu-calculus used in the timed setting is to add freeze quantification; see e.g., [9]. The extension of our proof rules to this setting should be similarly straightforward as the extension we presented in this article, although the definitions of the underlying transition systems would need extension to accommodate clock variables explicitly.

APPENDICES

A PROOFS FOR SECTION 4

We give the proof of Lemma 4.3: given monotonic function f over the subset lattice generated by set S, and a set $\mathcal{W} \subseteq 2^S$ of well-supported sets for f, set $\bigcup \mathcal{W}$ is well-supported for f.

The idea of the proof is as follows. We first order $\mathcal{W}$ arbitrarily, but in a well-founded manner. A well-founded relation for $\bigcup \mathcal{W}$ is then constructed by taking the elements that are less that $x \in \bigcup \mathcal{W}$ from the first set in $\mathcal{W}$ containing x according to the ordering. The resulting relation is then shown to induce a well-founded support structure on $\bigcup \mathcal{W}$.

The proof itself uses constructions over the ordinals, for which we use the von Neumann definition [16]: a well-ordered set α is a von Neumann ordinal iff it contains all ordinals preceding α. The well-ordering on von Neumann ordinals is often written $<$ and has the property that $\alpha < \beta$ iff $\alpha \in \beta$, which in turn is true iff $\alpha \subsetneq \beta$. Recall that in this article, we assume the Axiom of Choice [37].

Proof of Lemma 4.3. To prove that $\bigcup \mathcal{W}$ is well-supported for f, we first note that there is an ordinal β that is in bijective correspondence with $\mathcal{W}$. We then define ordinal-indexed sequences, X_α and $\prec_\alpha$ ($\alpha < \beta$), with the property that $(X_\alpha, \prec_\alpha)$ is a support structure for f, $\prec_\alpha$ is well-founded, and $\bigcup \mathcal{W} = \bigcup_{\alpha<\beta} X_\alpha$. We subsequently show that $\prec = \bigcup_{\alpha<\beta} \prec_\alpha$ is well-founded and that $(\bigcup \mathcal{W}, \prec)$ is a support structure for f, thereby establishing that $\bigcup \mathcal{W}$ is well-supported for f.

To this end, let $|\mathcal{W}| = \beta$ be the cardinality of $\mathcal{W}$ (i.e., β is the least ordinal in bijective correspondence to $\mathcal{W}$), and let $h \in \beta \to \mathcal{W}$ be a bijection. It follows that $\mathcal{W} = \{h(\alpha) \mid \alpha < \beta\}$. Also let $o \in \beta \to 2^{S \times S}$ be such that for any $\alpha < \beta$, $o(\alpha) \subseteq h(\alpha) \times h(\alpha)$ is a well-founded binary relation with the property that $(h(\alpha), o(\alpha))$ is a support structure for f.

We now define the following ordinal-indexed sequences X_α, X'_α and $\prec_\alpha$, $\alpha < \beta$, of subsets of S and binary relations on X_α, respectively, as follows, using transfinite recursion.

$$\begin{aligned}
X_\alpha &= \Big(\bigcup_{\alpha'<\alpha} X_{\alpha'}\Big) \cup h(\alpha),\\
X'_\alpha &= h(\alpha) \setminus \Big(\bigcup_{\alpha'<\alpha} X_{\alpha'}\Big),\\
\prec_\alpha &= \Big(\bigcup_{\alpha'<\alpha} \prec_{\alpha'}\Big) \cup \Big\{(x', x) \in o(\alpha) \mid x \in X'_\alpha\Big\}.
\end{aligned}$$

Note that $X_\alpha = (\bigcup_{\alpha'<\alpha} X_{\alpha'}) \cup X'_\alpha$ and that $(\bigcup_{\alpha'<\alpha} X_{\alpha'}) \cap X'_\alpha = \emptyset$. Based on these definitions, it is easy to see that if $\alpha' < \alpha$ then $X_{\alpha'} \subseteq X_\alpha$ and $\prec_{\alpha'} \subseteq \prec_\alpha$. We now prove two properties of X_α and $\prec_\alpha$ that will be used in what follows.

P1. For all $\alpha < \beta$, $(X_\alpha, \prec_\alpha)$ is a support structure over f.
P2. For all $\alpha < \beta$, $\prec_\alpha$ is well-founded.

To prove P1 we use transfinite induction. So fix $\alpha < \beta$; the induction hypothesis states that for any $\alpha' < \alpha$, $(X_{\alpha'}, \prec_{\alpha'})$ is a support structure over f. Now consider $x \in X_\alpha$; we must show that

$x \in f(\prec_{\alpha}^{-1}(x))$. There are two cases. In the first, $x \in \bigcup_{\alpha' < \alpha} X_{\alpha'}$, which means $x \in X_{\alpha'}$ for some $\alpha' < \alpha$. In this case the induction hypothesis guarantees that $(X_{\alpha'}, \prec_{\alpha'})$ is a support structure, meaning $x \in f(\prec_{\alpha'}^{-1}(x))$. Since $\prec_{\alpha'} \subseteq \prec_{\alpha}$ it follows that

$$\prec_{\alpha'}{}^{-1}(x) \subseteq \prec_{\alpha}{}^{-1}(x),$$

and since f is monotonic and $x \in f(\prec_{\alpha'}^{-1}(x))$, also $x \in f(\prec_{\alpha}^{-1}(x))$. In the second case, $x \in X'_{\alpha}$. Here it is easy to see that

$$\prec_{\alpha}{}^{-1}(x) = \{x' \mid (x', x) \in o(\alpha)\},$$

and since $(h(\alpha), o(\alpha))$ is a support structure, it immediately follows that $x \in f(\{x' \mid (x', x) \in o(\alpha)\}) = f(\prec_{\alpha}^{-1}(x))$. P1 is thus proved.

To prove P2 we again use transfinite induction. So fix $\alpha < \beta$. The induction hypothesis states that for all $\alpha' < \alpha$, $\prec_{\alpha'}$ is well-founded; we must show that $\prec_{\alpha}$ is as well. So consider a descending chain $C = \cdots \prec_{\alpha} x_2 \prec_{\alpha} x_1$; it suffices to show that C must be finite. There are three cases to consider.

C is a chain in $\bigcup_{\alpha' < \alpha} \prec_{\alpha'}$. In this case $x_1 \in \bigcup_{\alpha' < \alpha} X_{\alpha'}$, meaning there is an $\alpha' < \alpha$ such that $x_1 \in X_{\alpha'}$. Since $\alpha_1 < \alpha_2$ implies $\prec_{\alpha_1} \subseteq \prec_{\alpha_2}$, it follows that each $x_i \in X_{\alpha'}$, each $x_{i+1} \prec_{\alpha'} x_i$, and that C is thus a descending chain in $\prec_{\alpha'}$. Since the induction hypothesis guarantees that $\prec_{\alpha'}$ is well-founded, C must be finite.

C is a chain in $o(\alpha)$. In this case, since $o(\alpha)$ is well-founded, C must be finite.

C is a mixture of $\bigcup_{\alpha' < \alpha} \prec_{\alpha'}$ and $o(\alpha)$. In this case, from the definition of C and $\prec_{\alpha}$ it follows that C can be split into two pieces:

(1) an initial segment $x_i \prec_{\alpha} \cdots \prec_{\alpha} x_1$, where $i \geq 1$, $x_i \in \bigcup_{\alpha' < \alpha} X_{\alpha'}$, and for all $i > j \geq 1$, $x_j \in X'_{\alpha}$ and $(x_{j+1}, x_j) \in o(\alpha)$; and
(2) a segment $\cdots \prec_{\alpha} x_{i+1} \prec_{\alpha} x_i$, where for all $j \geq i$, $(x_{j+1}, x_j) \in \bigcup_{\alpha' < \alpha} \prec_{\alpha'}$.

The previous arguments establish that each of these sub-chains must be finite, and thus C is finite as well.

To finish the proof of the lemma, we note that the following hold, using arguments given above.

- $(\bigcup_{\alpha < \beta} X_{\alpha}, \bigcup_{\alpha < \beta} \prec_{\alpha})$ is a support structure.
- $\bigcup_{\alpha < \beta} \prec_{\alpha}$ is well-founded.
- $\bigcup \mathcal{W} = \bigcup_{\alpha < \beta} X_{\alpha}$.

From the definitions it therefore follows that $\bigcup \mathcal{W}$ is well-supported. □

B PROOFS FOR SECTION 6

We prove pseudo-transitivity of the dependency ordering relations. That is, (1) if $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_2} s_2$ and $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$ then $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_1} s_1$; (2) if $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ and $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$ then $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$; and (3) if $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ and $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_1$ then $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.

Proof of Lemma 6.19. We prove the three parts separately.

(1) Assume $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_2} s_2$. The result follows by induction on the definition of $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$.
(2) Assume that $s_2 \lessdot_{\mathbf{n}_2,\mathbf{n}_1} s_1$. The proof proceeds by induction on the definition of $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$. There are two cases to consider.
  - $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_2} s_2$. From the pseudo-transitivity of $\lessdot_{\mathbf{n}',\mathbf{n}}$ (first part of this lemma) it follows that $s_3 \lessdot_{\mathbf{n}_3,\mathbf{n}_1} s_1$, and thus $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.
  - There exists companion node $\mathbf{m}$, with $\mathbf{m} \neq \mathbf{n}_2$ and $\mathbf{m} \neq \mathbf{n}_3$, and $t, t' \in st(\mathbf{m})$ such that $s_3 <:_{\mathbf{n}_3,\mathbf{m}} t'$, $t' <:_{\mathbf{m}}^{+} t$ and $t \lessdot_{\mathbf{m},\mathbf{n}_2} s_2$. The first part of this lemma ensures that $t \lessdot_{\mathbf{m},\mathbf{n}_1} s_1$, and the definition of $<:_{\mathbf{n}_3,\mathbf{n}_1}$ confirms $s_3 <:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.

(3) Assume that $s_3 <:_{\mathbf{n}_3,\mathbf{n}_2} s_2$. The result then follows by induction on the definition of $s_2 <:_{\mathbf{n}_2,\mathbf{n}_1} s_1$, using the previous part of this lemma. □

C ADDITIONAL RESULTS AND PROOFS FOR SECTION 7

We first establish a technical property, derived from the definition of $<:_{\mathbf{m},\mathbf{n}}$, that is satisfied by $\leq:_{\mathbf{m},\mathbf{n}}$.

Lemma C.1 (Characterization of $\leq:_{\mathbf{m},\mathbf{n}}$). *Let $\mathbf{n}_1, \mathbf{n}_2$, and $\mathbf{n}_3$ be proof nodes in $\mathbb{T}$, with $\mathbf{n}_2$ a companion node and $\mathbf{n}_3 \neq \mathbf{n}_2$. If $s_2 \leq:_{\mathbf{n}_2,\mathbf{n}_1} s_1$, $s_2' <:^{+}_{\mathbf{n}_2} s_2$, and $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_2} s_2'$, then $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_1} s_1$.*

Proof. Follows from the definitions of $\leq:_{\mathbf{m},\mathbf{n}}$ and $<:_{\mathbf{m},\mathbf{n}}$ and pseudo-transitivity (Lemma 6.19) of $<:_{\mathbf{m},\mathbf{n}}$. □

We next use this property to show that $\leq:_{\mathbf{m},\mathbf{n}}$ obeys a pseudo-transitivity law (Lemma 7.11). That is, if $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_2} s_2$ and $s_2 \leq:_{\mathbf{n}_2,\mathbf{n}_1} s_1$, then $s_3 \leq:_{\mathbf{n}_3,\mathbf{n}_1} s_1$

Proof of Lemma 7.11. Follows from pseudo-transitivity of $<:_{\mathbf{m},\mathbf{n}}$ (Lemma 6.19) and the preceding lemma. □

D ADDITIONAL RESULTS AND PROOFS FOR SECTION 8

We first give a proof of Lemma 8.5, which states that if $R \subseteq S \times S$ is total and quotient well-founded, then R is a quotient well-ordering.

Proof of Lemma 8.5. Let $R \subseteq S \times S$ be total and qwf, with $(\mathcal{Q}_R, \sqsubseteq)$ the quotient of R. We must show that $\sqsubset\ =\ \sqsubseteq^{-}$ is a well-ordering, i.e., is well-founded and total, over $\mathcal{Q}_R$. Well-foundedness of $\sqsubset$ is immediate from the fact that R is qwf. We now must show that $\sqsubset$ is total, i.e., is irreflexive and transitive and has the property that for any $Q_1, Q_2 \in \mathcal{Q}_R$ such that $Q_1 \neq Q_2$, either $Q_1 \sqsubset Q_2$ or $Q_2 \sqsubset Q_1$. Irreflexivity and transitivity are immediate from the fact that $\sqsubset = \sqsubseteq^{-}$ is the irreflexive core of partial order $\sqsubseteq$. Now suppose $Q_1, Q_2 \in \mathcal{Q}_R$ are such that $Q_1 \neq Q_2$; we must show that either $Q_1 \sqsubset Q_2$ or $Q_2 \sqsubset Q_1$. From the definition of $\mathcal{Q}_R$ it follows that there are $s_1, s_2 \in S$ such that $Q_1 = [s_1]_R$ and $Q_2 = [s_2]_R$. Moreover, since $Q_1 \neq Q_2$ it must be that $s_1 \not\sim_R s_2$, and since R is total we have that either $s_1\ R\ s_2$ and $s_2\ \not R s_1$, whence $Q_1 = [s_1]_R \sqsubset [s_2]_R = Q_2$, or $s_2\ R\ s_1$ and $s_1\ \not R s_2$, whence $Q_2 \sqsubset Q_1$. As $\sqsubset$ is a well-ordering on $\mathcal{Q}_R$, by definition R is a qwo. □

Next we give the proof of Lemma 8.7, which states that if $R \subseteq S \times S$ is a total quotient well-ordering, then every non-empty subset $X \subset S$ contains an R-pseudo-minimum element.

Proof of Lemma 8.7. Fix total qwo $R \subseteq S \times S$, and let $X \subseteq S$ be non-empty. We must exhibit an R-pseudo-minimum element $x \in X$. To this end, consider the quotient $(\mathcal{Q}_R, \sqsubseteq)$ of R, and let $\sqsubset\ =\ \sqsubseteq^{-}$ be the irreflexive core of $\sqsubseteq$. Note that $\sqsubset$ is a well-ordering over $\mathcal{Q}_R$. Now consider $\mathcal{Q}_X \subseteq \mathcal{Q}_R$ defined by $\mathcal{Q}_X = \{\,[x]_R \mid x \in S\}$. It follows that there is a $Q \in \mathcal{Q}_X$ that is a $\sqsubset$-minimum for $\mathcal{Q}_X$, and that $Q = [x]_R$ for some $x \in X$. Since R is total it is transitive, meaning $R^* = R^{=}$. Then $x\ R\ x'$ for all $x' \in [x]_R$ such that $x \neq x'$; it is also the case that $x\ R\ x'$ for all x' such that $[x]_R \sqsubset [x']_R$. These facts imply that $x\ R\ x'$ for all $x' \in X$. □

Finally, let f, g be monotonic functions over the subset lattice induced by S, and $\sigma_1, \sigma_2 \in \{\mu, \nu\}$ with $X = \sigma_1(f[\sigma_2]g)$. Also, let $(X, \prec)$ be a σ_1-compatible, total qwf support structure for $f[\sigma_2]g$ and $Y = \sigma_2 g_{(\prec^{-1}(X),\cdot)}$. We show there is a σ_2-compatible, total qwf support structure $(Y, \prec')$ for $g_{(\prec^{-1}(X),\cdot)}$ that is locally consistent with $(X, \prec)$ (Lemma 8.9).

Proof of Lemma 8.9. Fix monotonic $f, g \in 2^S \times 2^S \to 2^S$ and $\sigma_1, \sigma_2 \in \{\mu, \nu\}$, let $X = \sigma_1(f[\sigma_2]g)$, and let $\prec \subseteq X \times X$ be such that $(X, \prec)$ is a σ_1-compatible, total qwf support structure for $f[\sigma_2]g$. Also fix $Y = \sigma_2 g_{(\prec^{-1}(X), \cdot)}$. We must construct a σ_2-compatible, total qwf $\prec' \subseteq Y \times Y$ such that $(Y, \prec')$ is a support structure for $g_{(\prec^{-1}(X), \cdot)}$ that is locally consistent with $(X, \prec)$.

Let $(\mathcal{Q}_\prec, \sqsubseteq)$ be the quotient of $(S, \prec)$ as given in Definition 3.8, and let $\sqsubset = \sqsubseteq^-$ be the irreflexive core of $\sqsubseteq$. For notational convenience, if $Z \subseteq S$ then we define

$$g_Z = g_{(\prec^{-1}(Z), \cdot)}.$$

Since $\prec$ is total and qwf it follows that $\sqsubset$ is a well-ordering on $\mathcal{Q}_\prec$. For any $Q \in \mathcal{Q}_\prec$ define

$$Y_Q = \sigma_2\, g_Q,$$

and let $(Y_Q, \prec'_Q)$ be a σ_2-maximal support structure for g_Q. Since $\prec'_Q \subseteq Y_Q \times Y_Q$ is σ_2-maximal, we have that $\prec'_Q$ is a well-ordering if $\sigma_2 = \mu$, and $U_{Y_Q} = Y_Q \times Y_Q$ if $\sigma_2 = \nu$. In either case it is easy to see that $\prec'_Q$ is total and qwf. Moreover, since $Q \subseteq X$ it follows that $\prec^{-1}(Q) \subseteq \prec^{-1}(X)$, and this means that for all $Z \subseteq S$, $g_Q(Z) \subseteq g_X(Z)$. Consequently $(Y_Q, \prec'_Q)$ is also a σ_2-compatible support structure for g_X, as for all $y \in Y_Q$, $y \in g_Q({\prec'_Q}^{-1}(y)) \subseteq g_X({\prec'_Q}^{-1}(y))$. We now define the following using well-founded induction on $\sqsubset = P(\prec)^-$.

$$\begin{aligned}
Y'_{\sqsubset Q} &= \bigcup_{Q' \sqsubset Q} Y'_{Q'}, \\
Y'_Q &= Y_Q \cup Y'_{\sqsubset Q}, \\
Y''_Q &= Y_Q \setminus Y'_{\sqsubset Q}, \\
\prec''_Q &= \left(\bigcup_{Q' \sqsubset Q} \prec''_{Q'} \right) \cup \left(Y'_{\sqsubset Q} \times Y''_Q \right) \cup \left(\prec'_Q \lfloor Y''_Q \right).
\end{aligned}$$

An inductive argument establishes that for each Q, $(Y'_Q, \prec''_Q)$ is a σ_2-compatible support structure for g_Q and that $\prec''_Q$ is total and qwf.

Now consider $Y' = \bigcup_{Q \in \mathcal{Q}_\prec} Y'_Q$ and $\prec'' = \bigcup_{Q \in \mathcal{Q}_\prec} \prec''_Q$. It is straightforward to show that $(Y', \prec'')$ is a σ_2-compatible, total, qwf support structure for g_X since each $(Y_Q, \prec'_Q)$ is. Also note that $Y' \subseteq Y$. To finish the construction of $(Y, \prec')$, take $Y'' = Y \setminus Y'$, and let $(Y, \prec''')$ be a maximal σ_2-compatible support structure for g_X. Note that $\prec'''$ is qwf, and well-founded if $\sigma_2 = \mu$ and U_Y if $\sigma_2 = \nu$. Now define the following:

$$\prec' = \prec'' \cup (Y' \times Y'') \cup (\prec''' \lfloor Y'').$$

From the reasoning above it follows that $(Y, \prec')$ is a σ_2-compatible support structure for g_X, and that $\prec'$ is total and qwf.

To complete the proof we must show that $(Y, \prec')$ is locally consistent with $(X, \prec)$. To this end, fix $x \in X$ and define $\prec'_x = (\prec')\lfloor Y_x$. We must show that $(Y_x, \prec'_x)$ is a σ_2-compatible support structure for $g_x = g_{\{x\}}$. Recall that $[x] \in \mathcal{Q}_\prec$ is the equivalence class containing x. We begin by noting that since $\prec$ is total and qwf,

$$\prec^{-1}(x) = [x] \cup \left(\bigcup_{Q \sqsubset [x]} Q \right).$$

Also, $\sigma_2(g_x) = \sigma_2(g_{[x]}) = Y_{[x]}$. The definition of $\prec'$ further guarantees that $\prec'_x = \prec''_{Q_x}$. As we know that $(Y_Q, \prec''_Q)$ is a σ_2-compatible support structure for g_Q for all $Q \in \mathcal{Q}_\prec$, the desired result holds. □

We next proved the detailed proof showing that if $S \subseteq ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$ then there is successful TNF tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi$.

Proof of Lemma 8.13. Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be an LTS of sort Σ and $\mathcal{V}$ be a valuation. The proof proceeds by structural induction on Φ; the induction hypothesis states for any subformula Φ' of Φ and S' such that $S' \subseteq ||\,\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$, that $S' \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi'$ has a successful TNF tableau. The argument involves a case analysis on the form of Φ. Most cases are routine and left to the reader. We consider here the case for $\langle K\rangle$.

So assume $\Phi = \langle K\rangle\Phi'$, and let $f \in S \rightarrow \mathcal{S}$ be a function such that for every $s \in S$, $s \xrightarrow{K} f(s)$ and $f(s) \in ||\,\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$. Such an f must exist, as $S \subseteq ||\,\langle K\rangle\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$ and thus for every $s \in S$ there is an $s' \in \mathcal{S}$ such that $s \xrightarrow{K} s'$ and $s' \in ||\,\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$. Since $f(S) \subseteq ||\,\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$, the induction hypothesis guarantees the existence of a successful TNF tableau for $f(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi'$. We now construct a successful TNF tableau for $S \vdash^{\mathcal{T},\mathcal{V}'}_{\varepsilon} \Phi$ as follows. Create a fresh tree node labeled by $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi$, and let its only child be the root node for the successful TNF tableau for $f(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi'$. The rule application associated with the new node is $(\langle K\rangle, f)$. The new tableau is clearly successful and TNF. □

E ADDITIONAL RESULTS AND PROOFS FOR SECTION 10

We first establish completeness for timed mu-calculus formulas without fixpoints, extending the result in Lemma 8.13.

Lemma E.1 (Timed Fixpoint-free Completeness). *Let $\mathcal{T}, \mathcal{V}, \Phi$, and S be such that Φ is a fixpoint-free timed mu-calculus formula and $S \subseteq ||\,\Phi\,||^{\mathcal{T}}_{\mathcal{V}}$. Then there is a successful TNF timed tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi$.*

Proof. Let $\mathcal{T} = (\mathcal{S},\rightarrow)$ be a TTS of timed sort Σ, and $\mathcal{V}$ be a valuation. The proof proceeds by structural induction on Φ; the induction hypothesis states that for any subformula Φ' of Φ and S', if $S' \subseteq ||\,\Phi'\,||^{\mathcal{T}}_{\mathcal{V}}$ then $S' \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi'$ has a successful TNF timed tableau. The proof is completely analogous to that of Lemma 8.13, and involves a case analysis on the form of Φ. We here only show the cases for the new operators, $\exists$ and $\forall$.

Assume $\Phi = \exists_{\Phi_1}\Phi_2$; we first establish that there exists a function $f \in S \rightarrow \mathbb{R}_{\geq 0}$ such that for all $s \in S$, $f(s) \in del(s)$, $succ_{<}(s, f(s)) \subseteq ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}}$, and $succ(s, f(s)) \in ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$. Fix arbitrary $s \in S$. Then $s \in ||\,\exists_{\Phi_1}\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$, and the definition of $||\,\exists_{\Phi_1}\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$ guarantees the existence of $\delta \in del(s)$ such that $succ_{<}(s,\delta) \subseteq ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}}$ and $succ(s,\delta) \in ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$. Let δ be such and set $f(s) = \delta$. This immediately satisfies the required conditions. It follows from this definition of f that $f_{<}(S) \subseteq ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}}$. Furthermore, as for every $s \in S$, $succ(s, f(s)) \in ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$, also $f_{=}(S) \subseteq ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$. Hence, the induction hypothesis guarantees that successful TNF timed tableaux exist for $f_{<}(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_1$ and $f_{=}(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_2$. Without loss of generality, assume that these tableaux have disjoint sets of proof nodes. We now construct a successful TNF timed tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \exists_{\Phi_1}\Phi_2$ as follows. Create a fresh tree node labeled by $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \exists_{\Phi_1}\Phi_2$, having as its left child the root node of the successful TNF timed tableau for $f_{<}(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_1$ and as its right child the root of the successful TNF timed tableau for $f_{=}(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_2$. The rule application associated with the new node is $(\exists, f)$. The new tableau is clearly successful and TNF.

Now assume $\Phi = \forall_{\Phi_1}\Phi_2$; we first establish that there exists a function $g \in S \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ such that, for all $s \in S$, $\delta \in del(s)$, $g(s,\delta) \leq \delta$, and such that if $g(s,\delta) = \delta$, $succ(s, g(s,\delta)) \in ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$, and $succ(s, g(s,\delta)) \in ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}}$, otherwise. Fix arbitrary $s \in S$. Since $s \in ||\,\forall_{\Phi_1}\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$, the definition of $||\,\forall_{\Phi_1}\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$ guarantees that, for all $\delta \in del(s)$, either $succ_{<}(s,\delta) \cap ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}} \neq \emptyset$ or $succ(s,\delta) \in ||\,\Phi_2\,||^{\mathcal{T}}_{\mathcal{V}}$. So, if $succ_{<}(s,\delta) \cap ||\,\Phi_1\,||^{\mathcal{T}}_{\mathcal{V}} \neq \emptyset$ there is some $\delta' < \delta$ such that $succ(s,\delta') \in$

$|| \Phi_1 ||^{\mathcal{T}}_{\mathcal{V}}$, and we choose $g(s,\delta) = \delta'$. Otherwise, $succ(s,\delta) \in || \Phi_2 ||^{\mathcal{T}}_{\mathcal{V}}$ and we can choose $g(s,\delta) = \delta$. Given such a g, it is easy to see that $g_<(S) \subseteq || \Phi_1 ||^{\mathcal{T}}_{\mathcal{V}}$ and $g_=(S) \subseteq || \Phi_2 ||^{\mathcal{T}}_{\mathcal{V}}$; hence the induction hypothesis guarantees that successful TNF timed tableaux exist for $g_<(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_1$ and $g_=(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_2$. Without loss of generality, assume that these tableaux have disjoint sets of proof nodes. We now construct a successful TNF timed tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \forall_{\Phi_1}\Phi_2$ as follows. Create a fresh tree node labeled by $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \forall_{\Phi_1}\Phi_2$, having as its left child the root node of the successful TNF timed tableau for $g_<(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_1$ and as its right child the root of the successful TNF timed tableau for $g_=(S) \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \Phi_2$. The rule application associated with the new node is $(\forall, g)$. The new tableau is clearly successful and TNF. □

We next establish the existence of successful TNF timed tableaux for valid sequents in the case where formulas have the form $\sigma Z.\Phi$, where Φ does not contain fixpoint subformulas, and have specific σ-compatible fixpoint orderings, analogous to Lemma 8.15. The proof, in particular, shows in detail how to merge tableaux containing rules $\forall$ and $\exists$.

Lemma E.2 (Timed Single-fixpoint Completeness). *Let $\mathcal{T}$ be a TTS, and let Φ, Z, $\mathcal{V}$, σ and S be such that Φ is a fixpoint-free timed mu-calculus formula and $S = || \sigma Z.\Phi ||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $|| Z.\Phi ||^{\mathcal{T}}_{\mathcal{V}}$. Then $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ has a successful TNF timed tableau compliant with $(S, \prec)$.*

Proof. Fix TTS $\mathcal{T} = (\mathcal{S}, \rightarrow)$ of sort Σ, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that Φ is a fixpoint-free timed mu-calculus formula, and $S = || \sigma Z.\Phi ||^{\mathcal{T}}_{\mathcal{V}}$. Also let $(S, \prec)$ be a σ-compatible, total, qwf support structure for $f_\Phi = || Z.\Phi ||^{\mathcal{T}}_{\mathcal{V}}$. We must construct a successful TNF timed tableau for sequent $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ that is compliant with $(S, \prec)$. The proof mirrors the one given for Lemma 8.15 and consists of the following steps:

(1) For each $s \in S$ we use Lemma E.1 to establish the existence of a successful TNF timed tableau for sequent $\{s\} \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$, where $\mathcal{V}_s = \mathcal{V}[Z := \prec^{-1}(s)]$.
(2) We then construct a successful TNF timed tableau for sequent $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$, where $\mathcal{V}_S = \mathcal{V}[Z := \prec^{-1}(S)]$, from the individual tableaux for the $s \in S$.
(3) We convert the tableau for $S \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$ into a successful TNF timed tableau for $S \vdash^{\mathcal{T},\mathcal{V}}_{\varepsilon} \sigma Z.\Phi$ that is compliant with $\prec$.

The proofs of steps 1 and 3 are completely analogous to the one in the proof of Lemma 8.15, using Lemma E.1 instead of Lemma 8.13. We here focus on Step 2 of the proof outline, and construct a tableau for $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$.

Let $\mathbb{T}_s = (\mathcal{T}, \mathcal{V}_s, \mathbf{T}, \rho_s, \lambda_s)$ be the successful TNF timed tableau whose root is $\{s\} \vdash^{\mathcal{T},\mathcal{V}_s}_{\varepsilon} \Phi$, with $\mathbf{T} = (\mathbf{N}, \mathbf{r}, p, cs)$ the common tree shared by all these structurally equivalent tableaux, with $fm(\mathbf{n})$ and $rn(\mathbf{n})$ for $\mathbf{n} \in \mathbf{N}$ the common formulas and rule names each tableau includes in $\mathbf{n}$.

We now adapt the construction in the proof of Lemma 8.15 to build a successful TNF timed tableau for $S \vdash^{\mathcal{T},\mathcal{V}_S}_{\varepsilon} \Phi$ satisfying the following: if s, s' and $\mathbf{n}'$ are such that $fm(\mathbf{n}') = Z$ and $s' <:_{\mathbf{n}',\mathbf{r}} s$, then $s' \prec s$. There are two cases to consider. The proof if $S = \emptyset$ is analogous to that in Lemma 8.15. So, assume $S \neq \emptyset$. We will construct $\mathbb{T}_S = (\mathcal{T}, \mathcal{V}_S, \mathbf{T}, \rho_S, \lambda_S)$ that is structurally equivalent to each $\mathbb{T}_s$ for $s \in S$. As was the case in the proof of Lemma 8.15 the idea is to appropriately "merge" the individual tableaux $\mathbb{T}_s$ for the $s \in S$. The construction uses a co-inductive strategy to define ρ_S and λ_S so that invariants I1–I3 in Lemma 8.15 hold for $\mathbf{n} \in \mathbf{N}$. Specifically, $\lambda_S(\mathbf{r})$ is set to be sequent $S \vdash^{\mathcal{T},\varepsilon}_{\mathcal{V}_S} \Phi$; this ensures that invariants I2 and I3 hold of $\mathbf{r}$. Then, for every internal node $\mathbf{n}$ for which $\lambda_S(\mathbf{n})$ has been defined and for which invariants I2 and I3 hold, $\rho_S(\mathbf{n})$ and $\lambda_S(\mathbf{n}')$ for each child $\mathbf{n}'$ of $\mathbf{n}$ are defined so that invariant I1 holds of $\mathbf{n}$ and invariants I2 and I3 hold of each $\mathbf{n}'$. This processing of internal nodes is done using a case analysis on $rn(\mathbf{n})$. Let $S_{\mathbf{n}} = st(\lambda_S(\mathbf{n}))$ be the set of states in the sequent labeling $\mathbf{n}$. We only consider the cases where $rn(\mathbf{n}) \in \{\forall, \exists\}$; the other cases are identical to those in Lemma 8.15.

***rn*$(\mathbf{n}) = \exists$.** In this case, $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ and $fm(\mathbf{n}) = \exists_{fm(\mathbf{n}_1)} fm(\mathbf{n}_2)$. We begin by constructing a function $f_{\mathbf{n}} \in S_{\mathbf{n}} \to \mathbb{R}_{\geq 0}$ such that for all $s \in S_{\mathbf{n}}$, $f_{\mathbf{n}}(s) \in del(s)$, and also such that $(f_{\mathbf{n}})_<(S_{\mathbf{n}}) \subseteq \bigcup_{t \in S} st(\lambda_t(\mathbf{n}_1))$ and $(f_{\mathbf{n}})_=(S_{\mathbf{n}}) \subseteq \bigcup_{t \in S} st(\lambda_t(\mathbf{n}_2))$. This function will then be used to define $\rho_S(\mathbf{n}, \lambda_S(\mathbf{n}_1)$ and $\lambda_S(\mathbf{n}_2)$. So fix $s \in S_{\mathbf{n}}$; we construct $f_{\mathbf{n}}(s)$ based on the tableaux $\mathbb{T}_t$ whose sequence for $\mathbf{n}$ contains s. To this end, define

$$I_s = \{t \in S \mid s \in st(\lambda_t(\mathbf{n}))\}.$$

Intuitively, $I_s \subseteq S$ contains all states t whose tableau $\mathbb{T}_t$ contains state s in $\mathbf{n}$. Clearly I_s is non-empty and thus contains a pseudo-minimum element t (Lemma 8.7). Let $f_{\mathbf{n},t} \in st(\lambda_t(\mathbf{n})) \to \mathbb{R}_{\geq 0}$ be such that $\rho_t(\mathbf{n}) = (\exists, f_{\mathbf{n},t})$. We know that for all $s \in st(\lambda_t(\mathbf{n}))$, $f_{\mathbf{n},t}(s) \in del(s)$, $succ_<(s, f_{\mathbf{n},t}(s)) \subseteq st(\lambda_t(\mathbf{n}_1))$, and $\{succ(s, f_{\mathbf{n},t}(s)) \mid s \in st(\lambda_t(\mathbf{n}))\} \subseteq st(\lambda_t(\mathbf{n}_2))$. We now define $f_{\mathbf{n}}(s) = f_{\mathbf{n},t}(s)$ Finally, we define $\rho_S(\mathbf{n}) = (\exists, f_{\mathbf{n}})$, $\lambda_S(\mathbf{n}_1) = (f_{\mathbf{n}})_<(S_{\mathbf{n}}) \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} fm(\mathbf{n}_1)$, and $\lambda_S(\mathbf{n}_2) = (f_{\mathbf{n}})_=(S_{\mathbf{n}}) \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} fm(\mathbf{n}_2)$. It can be seen that invariant I1 holds of $\mathbf{n}$ and that I2 and I3 hold of $\mathbf{n}_1$ and $\mathbf{n}_2$.

***rn*$(\mathbf{n}) = \forall$.** In this case, $cs(\mathbf{n}) = \mathbf{n}_1\mathbf{n}_2$ and $fm(\mathbf{n}) = \forall_{fm(\mathbf{n}_1)} fm(\mathbf{n}_2)$. We begin by constructing a function $g_{\mathbf{n}} \in S_{\mathbf{n}} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $(g_{\mathbf{n}})_<(S_{\mathbf{n}}) \subseteq \bigcup_{t \in S} st(\lambda_t(\mathbf{n}_1))$ and $(g_{\mathbf{n}})_=(S_{\mathbf{n}}) \subseteq \bigcup_{t \in S} st(\lambda_t(\mathbf{n}_2))$. This function will then be used to define $\rho_S(\mathbf{n})$, $\lambda_S(\mathbf{n}_1)$ and $\lambda_S(\mathbf{n}_2)$ so that the desired invariants hold. So fix $s \in S_{\mathbf{n}}$ and $\delta \in del(s)$; we construct $g_{\mathbf{n}}(s, \delta)$ based on the tableaux $\mathbb{T}_t$ whose sequent for $\mathbf{n}$ contains s. To this end, define

$$I_s = \{t \in S \mid s \in st(\lambda_t(\mathbf{n}))\}.$$

Intuitively, $I_s \subseteq S$ contains all states t whose tableau $\mathbb{T}_t$ contains state s in $\mathbf{n}$. Clearly I_s is non-empty and thus contains a pseudo-minimum element t with respect to $\prec$ (Lemma 8.7). Let $g_{\mathbf{n},t} \in st(\lambda_t(\mathbf{n})) \times \mathcal{S} \to \mathcal{S}$ be such that $\rho_t(\mathbf{n}) = (\forall, g_{\mathbf{n},t})$. We know that for all $\delta \in succ(s)$, if $g_{\mathbf{n},t}(s, \delta) < \delta$ then $succ(s, g_{\mathbf{n},t}(s, \delta)) \in st(\lambda_t(\mathbf{n}_1))$, and if $g_{\mathbf{n},t}(s, \delta) = \delta$ then $succ(s, g_{\mathbf{n},t}(s, \delta)) \in st(\lambda_t(\mathbf{n}_2))$. We take $g_{\mathbf{n}}(s, \delta) = g_{\mathbf{n},t}(s, \delta)$. Finally, we define $\rho(\mathbf{n}) = (\forall, g_{\mathbf{n}})$, $\lambda_S(\mathbf{n}_1) = (g_{\mathbf{n}})_<(S_{\mathbf{n}}) \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} fm(\mathbf{n}_1)$ and $\lambda_S(\mathbf{n}_2) = (g_{\mathbf{n}})_=(S_{\mathbf{n}}) \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}_S} fm(\mathbf{n}_2)$. It can be seen that invariant I1 holds of $\mathbf{n}$ and that I2 and I3 hold of $\mathbf{n}_1$ and $\mathbf{n}_2$.

This construction ensures that Properties I1–I3 hold for all $\mathbf{n}$.

To establish that $\mathbb{T}_S$ is successful we must show that every leaf in $\mathbb{T}_S$ is successful. The argument is identical to the proof in Lemma 8.15, Step 2. □

As an immediate corollary, we have the following:

Corollary E.3. *Fix $\mathcal{T}$, and let $\Phi, Z, \mathcal{V}, \sigma$ and S be such that Φ is a timed fixpoint-free formula and $S = \|\, \sigma Z.\Phi \,\|_{\mathcal{V}}^{\mathcal{T}}$. Then $S \vdash_{\varepsilon}^{\mathcal{T}, \mathcal{V}} \sigma Z.\Phi$ has a successful tableau.*

Proof. Follows from Lemma E.2 and the fact that every σ-maximal support structure $(S, \prec)$ for $\|\, Z.\Phi \,\|_{\mathcal{V}}^{\mathcal{T}}$ is total and qwf. □

REFERENCES

[1] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim G. Larsen. 2003. The power of reachability testing for timed automata. *Theoretical Computer Science* 300, 1 (2003), 411–475. DOI: https://doi.org/10.1016/S0304-3975(02)00334-1

[2] Luca Aceto, Augusto Burgueño, and Kim G. Larsen. 1998. Model checking via reachability testing for timed automata. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems.* Bernhard Steffen (Ed.), Lecture Notes in Computer Science, Springer, Berlin, 263–280. DOI: https://doi.org/10.1007/BFb0054177

[3] Luca Aceto and François Laroussinie. 2002. Is your model checker on time? On the complexity of model checking for timed modal logics. *The Journal of Logic and Algebraic Programming* 52–53, 58-1 (2002), 7–51. DOI: https://doi.org/10.1016/S1567-8326(02)00022-X

[4] Bahareh Afshari and Graham E. Leigh. 2017. Cut-free completeness for modal mu-calculus. In *Proceedings of the 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, Reykjavik, Iceland, 1–12. DOI: https://doi.org/10.1109/LICS.2017.8005088

[5] Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235. DOI: https://doi.org/10.1016/0304-3975(94)90010-8

[6] Henrik Reif Andersen. 1993. *Verification of Temporal Properties of Concurrent Systems*. Ph.D. Dissertation. Aarhus University, Denmark. DOI: https://doi.org/10.7146/dpb.v22i445.6762

[7] Yves Bertot and Pierre Castéran. 2013. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag, Berlin. DOI: https://doi.org/10.1007/978-3-662-07964-5

[8] Girish Bhat, Rance Cleaveland, and Orna Grumberg. 1995. Efficient on-the-fly model checking for CTL*. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 388–397. DOI: https://doi.org/10.1109/LICS.1995.523273

[9] Patricia Bouyer, Franck Cassez, and François Laroussinie. 2011. Timed modal logics for real-time systems. *Journal of Logic, Language, and Information* 20, 2 (2011), 169–203. DOI: https://doi.org/10.1007/s10849-010-9127-4

[10] Julian Bradfield and Colin Stirling. 1990. Verifying temporal properties of processes. In *Proceedings of the CONCUR'90 Theories of Concurrency: Unification and Extension*. J. C. M. Baeten and J. W. Klop (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 115–125. DOI: https://doi.org/10.1007/BFb0039055

[11] Julian Bradfield and Colin Stirling. 1992. Local model checking for infinite state spaces. *Theoretical Computer Science* 96, 1 (1992), 157–174. DOI: https://doi.org/10.1016/0304-3975(92)90183-G

[12] Julian Bradfield and Igor Walukiewicz. 2018. The mu-calculus and model checking. In *Proceedings of the Handbook of Model Checking*. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.), Springer International Publishing, Cham, 871–919. DOI: https://doi.org/10.1007/978-3-319-10575-8_26

[13] Julian C. Bradfield. 1991. *Verifying Temporal Properties of Systems with Applications to Petri Nets*. Ph.D. Dissertation. University of Edinburgh, Edinburgh. Retrieved from http://hdl.handle.net/1842/6565

[14] Julian C. Bradfield. 1993. A proof assistant for symbolic model-checking. In *Proceedings of the Computer Aided Verification*. Gregor von Bochmann and David Karl Probst (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 316–329. DOI: https://doi.org/10.1007/3-540-56496-9_25

[15] Florian Bruse, Oliver Friedmann, and Martin Lange. 2015. On guarded transformation in the modal μ-calculus. *Logic Journal of the IGPL* 23, 2 (2015), 194–216. DOI: https://doi.org/10.1093/jigpal/jzu030

[16] Krzysztof Ciesielski. 1997. *Set Theory for the Working Mathematician*. Cambridge University Press, Cambridge. DOI: https://doi.org/10.1017/CBO9781139173131

[17] Edmund M. Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. 2018. *Model Checking* (2nd. ed.). The MIT Press, Cambridge, Massachusetts.

[18] Rance Cleaveland. 1990. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica* 27, 8 (1990), 725–747. DOI: https://doi.org/10.1007/BF00264284

[19] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. 1993. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems* 15, 1 (1993), 36–72.

[20] Rance Cleaveland and Bernhard Steffen. 1993. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design* 2, 2 (1993), 121–147. DOI: https://doi.org/10.1007/BF01383878

[21] Robert L. Constable, Stuart F. Allen, H. M. Bromley, W. Rance Cleaveland, J. F. Cremer, Robert W. Harper, Doug J. Howe, Todd B. Knoblock, Nax P. Mendler, Prakash Panangaden, James T. Sasaki, and Scott F. Smith. 1986. *Implementing Mathematics*. Prentice-Hall, NJ.

[22] Sjoerd Cranen, Bas Luttik, and Tim A. C. Willemse. 2015. Evidence for fixpoint logic. In *Proceedings of the 24th EACSL Annual Conference on Computer Science Logic (CSL'15)*. Stephan Kreutzer (Ed.), Leibniz International Proceedings in Informatics, Vol. 41, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 78–93. DOI: https://doi.org/10.4230/LIPIcs.CSL.2015.78

[23] E. Allen Emerson and Charanjit S. Jutla. 1991. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium of Foundations of Computer Science*. IEEE Computer Society, 368–377. DOI: https://doi.org/10.1109/SFCS.1991.185392

[24] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. 2001. On model checking for the μ-Calculus and its fragments. *Theoretical Computer Science* 258, 1 (2001), 491–522. DOI: https://doi.org/10.1016/S0304-3975(00)00034-7

[25] Peter Fontana and Rance Cleaveland. 2014. The power of proofs: New algorithms for timed automata model checking. In *Proceedings of the Formal Modeling and Analysis of Timed Systems*. Springer, Cham, 115–129. DOI: https://doi.org/10.1007/978-3-319-10512-3_9

[26] Peter Christopher Fontana. 2014. *Towards a Unified Theory of Timed Automata.* Ph.D. Dissertation. University of Maryland, College Park. Retrieved from http://drum.lib.umd.edu/handle/1903/15232
[27] Oliver Friedmann and Martin Lange. 2013. Deciding the unguarded modal -calculus. *Journal of Applied Non-Classical Logics* 23, 4 (2013), 353–371. DOI: https://doi.org/10.1080/11663081.2013.861181
[28] Erich Grädel, Wolfgang Thomas, and Thomas Wilke (Eds.). 2002. *Automata, Logics, and Infinite Games: A Guide to Current Research [Outcome of a Dagstuhl Seminar, February 2001].* Lecture Notes in Computer Science, Vol. 2500. Springer, Berlin. DOI: https://doi.org/10.1007/3-540-36387-4
[29] Jan Friso Groote and Radu Mateescu. 1999. Verification of temporal properties of processes in a setting with data. In *Proceedings of the Algebraic Methodology and Software Technology.* Armando M. Haeberer (Ed.), Lecture Notes in Computer Science, Springer, Berlin, 74–90. DOI: https://doi.org/10.1007/3-540-49253-4_8
[30] Jan Friso Groote and Tim A. C. Willemse. 2005. Parameterised boolean equation systems. *Theoretical Computer Science* 343, 3 (2005), 332–369. DOI: https://doi.org/10.1016/j.tcs.2005.06.016
[31] Michiel Hazewinkel (Ed.). 2001. *Encyclopaedia of Mathematics, Supplement III.* Springer. 458 pages.
[32] Matthew Hennessy and Robin Milner. 1985. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM* 32, 1 (1985), 137–161. DOI: https://doi.org/10.1145/2455.2460
[33] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. 1994. Symbolic model checking for real-time systems. *Information and Computation* 111, 2 (1994), 193–244. DOI: https://doi.org/10.1006/inco.1994.1045
[34] Bart Jacobs and Jan Rutten. 2011. An introduction to (co)algebra and (co)induction. In *Proceedings of the Advanced Topics in Bisimulation and Coinduction.* Cambridge Tracts in Theoretical Computer Science, Vol. 52, Cambridge University Press, Cambride. DOI: https://doi.org/10.1017/CBO9780511792588.003
[35] Gerhard Jäger, Mathis Kretz, and Thomas Studer. 2008. Canonical completeness of infinitary μ. *The Journal of Logic and Algebraic Programming* 76, 2 (2008), 270–292. DOI: https://doi.org/10.1016/j.jlap.2008.02.005
[36] Thomas J. Jech. 1978. *Set Theory.* Academic Press, New York.
[37] Thomas J. Jech. 2008. *The Axiom of Choice.* Courier Corporation.
[38] Natthapong Jungteerapanich. 2009. A tableau system for the modal μ-calculus. In *Proceedings of the Automated Reasoning with Analytic Tableaux and Related Methods.* Martin Giese and Arild Waaler (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 220–234. DOI: https://doi.org/10.1007/978-3-642-02716-1_17
[39] Jeroen J. A. Keiren and Rance Cleaveland. 2022. Extensible proof systems for infinite-state systems. arXiv:2207.12953. Retrieved from https://arxiv.org/abs/2207.12953
[40] Alexander Kick. 1995. *Tableaux and Witnesses for the Mu–Calculus.* Technical Report. Fakultät für Informatik, Universität Karlsruhe, Karlsruhe.
[41] Nils Klarlund. 1990. *Progress Measures and Finite Arguments for Infinite Computations.* Ph.D. Dissertation. Cornell University. Retrieved from https://hdl.handle.net/1813/6993
[42] Nils Klarlund. 1992. Liminf progress measures. In *Proceedings of the Mathematical Foundations of Programming Semantics.* Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt (Eds.), Lecture Notes in Computer Science, Vol. 598. Springer, Berlin, 477–491. DOI: https://doi.org/10.1007/3-540-55511-0_25
[43] Nils Klarlund. 1992. Progress measures and stack assertions for fair termination. In *Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing (PODC'92).* ACM, New York, NY, 229–240. DOI: https://doi.org/10.1145/135419.135462
[44] Dexter Kozen. 1983. Results on the propositional μ-calculus. *Theoretical Computer Science* 27, 3 (1983), 333–354. DOI: https://doi.org/10.1016/0304-3975(82)90125-6
[45] Kenneth Kunen. 1980. *Set Theory: An Introduction to Independence Proofs.* Elsevier Science, Amsterdam, Lausanne, New York. Retrieved from http://opac.inria.fr/record=b1117475
[46] François Laroussinie, Kim G. Larsen, and Carsten Weise. 1995. From timed automata to logic — and back. In *Proceedings of the Mathematical Foundations of Computer Science 1995.* Jiří Wiedermann and Petr Hájek (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 529–539. DOI: https://doi.org/10.1007/3-540-60246-1_158
[47] Kim G. Larsen. 1988. Proof systems for hennessy-milner logic with recursion. In *Proceedings of the CAAP'88.* M. Dauchet and M. Nivat (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 215–230. DOI: https://doi.org/10.1007/BFb0026106
[48] Kim G. Larsen, Paul Pettersson, and Wang Yi. 1995. Model-checking for real-time systems. In *Proceedings of the Fundamentals of Computation Theory.* Horst Reichel (Ed.), Lecture Notes in Computer Science, Springer, Berlin, 62–88. DOI: https://doi.org/10.1007/3-540-60249-6_41
[49] Angelika Mader. 1993. Tableau recycling. In *Proceedings of the Computer Aided Verification.* Gregor von Bochmann and David Karl Probst (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 330–342. DOI: https://doi.org/10.1007/3-540-56496-9_26
[50] Angelika H. Mader. 1997. *Verification of Modal Properties Using Boolean Equation Systems.* Ph.D. Dissertation. Technische Universität München, München.

[51] Johannes Marti and Yde Venema. 2021. A focus system for the alternation-free μ-calculus. In *Proceedings of the Automated Reasoning with Analytic Tableaux and Related Methods.* Anupam Das and Sara Negri (Eds.), Lecture Notes in Computer Science, Springer International Publishing, Cham, 371–388. DOI: https://doi.org/10.1007/978-3-030-86059-2_22

[52] Radu Mateescu and Mihaela Sighireanu. 2003. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Science of Computer Programming* 46, 3 (2003), 255–281.

[53] Oleg V. Sokolsky and Scott A. Smolka. 1994. Incremental model checking in the modal mu-calculus. In *Proceedings of the International Conference on Computer Aided Verification (LNCS)*. Springer, Berlin, 351–363. DOI: https://doi.org/10.1007/3-540-58179-0_67

[54] Oleg V. Sokolsky and Scott A. Smolka. 1995. Local model checking for real-time systems. In *Proceedings of the International Conference on Computer Aided Verification (LNCS)*. Springer, Berlin, 211–224. DOI: https://doi.org/10.1007/3-540-60045-0_52

[55] Colin Stirling. 2001. *Modal and Temporal Properties of Processes.* Springer, New York, NY. DOI: https://doi.org/10.1007/978-1-4757-3550-5

[56] Colin Stirling. 2013. A proof system with names for modal mu-calculus. *Electronic Proceedings in Theoretical Computer Science* 129, 60-1 (2013), 18–29. DOI: https://doi.org/10.4204/EPTCS.129.2 arXiv:cs/1309.5129

[57] Colin Stirling. 2014. A tableau proof system with names for modal mu-calculus. In *Proceedings of the EPiC Series in Computing*. EasyChair, 306–318. DOI: https://doi.org/10.29007/lwqm

[58] Colin Stirling and David Walker. 1991. Local model checking in the modal mu-calculus. *Theoretical Computer Science* 89, 1 (1991), 161–177. DOI: https://doi.org/10.1016/0304-3975(90)90110-4

[59] Robert S. Streett and E. Allen Emerson. 1989. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation* 81, 3 (1989), 249–264. DOI: https://doi.org/10.1016/0890-5401(89)90031-X

[60] Thomas Studer. 2008. On the proof theory of the modal mu-calculus. *Studia Logica* 89, 3 (2008), 343–363. DOI: https://doi.org/10.1007/s11225-008-9133-6

[61] Li Tan and Rance Cleaveland. 2002. Evidence-based model checking. In *Proceedings of the Computer Aided Verification.* Ed Brinksma and Kim Guldstrand Larsen (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 455–470. DOI: https://doi.org/10.1007/3-540-45657-0_37

[62] Alfred Tarski. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5, 2 (1955), 285–309. Retrieved from https://projecteuclid.org/euclid.pjm/1103044538

[63] Igor Walukiewicz. 1993. On completeness of the mu -calculus. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*. IEEE, Montreal, QC, Canada, 136–146. DOI: https://doi.org/10.1109/LICS.1993.287593

[64] Igor Walukiewicz. 2000. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation* 157, 1 (2000), 142–182. DOI: https://doi.org/10.1006/inco.1999.2836

[65] Glynn Winskel. 1989. A note on model checking the modal V-Calculus. In *Proceedings of the Automata, Languages and Programming.* Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simonetta Ronchi Della Rocca (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 761–772. DOI: https://doi.org/10.1007/BFb0035797

[66] Dezhuang Zhang and Rance Cleaveland. 2005. Fast generic model-checking for data-based systems. In *Proceedings of the Formal Techniques for Networked and Distributed Systems - FORTE 2005*. Springer, Berlin, 83–97. DOI: https://doi.org/10.1007/11562436_8

[67] Dezhuang Zhang and Rance Cleaveland. 2005. Fast on-the-fly parametric real-time model checking. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS'05)*. IEEE, Miami, FL, 157–166. DOI: https://doi.org/10.1109/RTSS.2005.22

Received 20 September 2022; revised 24 March 2023; accepted 11 August 2023