Formalization of AMR Inference via Hybrid Logic Tableaux

Eli Goldner

July 29, 2021

Abstract

AMR and its extensions have become popular in semantic representation due to their ease of annotation by non-experts, attention to the predicative core of sentences, and abstraction away from syntactic matter. An area where AMR and its extensions warrant improvement is formalization and suitability for inference, where it is lacking compared to other semantic representations, such as description logics, episodic logic, and discourse representation theory. This thesis presents a formalization of inference over a merging of Donatelli et al.'s (2018) AMR extension for tense and aspect and with Pustejovsky et al.'s (2019) AMR extension for quantification and scope. Inference is modeled with a merging of Hansen's (2007) tableau method for first-order hybrid logic with varying domain semantics (*FHL*) and Blackburn and Jørgensen's (2012) tableau method for basic hybrid tense logic (*BHTL*). We motivate the merging of these AMR variants, present their interpretation and inference in the combination of *FHL* and *BHTL*, which we will call first-order hybrid tense logic (*FHTL*), and demonstrate the soundness and completeness of *FHTL*'s general tableau method, and its decidability for finite model checking.

1 Introduction

AMR's reductionist approach to semantic representation enjoys a duly earned popularity in the increasingly data driven environment of post-2000s NLP and CL. Its ablation of fine-grained semantic and syntactic features makes AMR a highly competitive choice for the development of annotated corpora and as a target representation for statistically driven parsing techniques. It is exactly the aspects of AMR that are so valued that cause it to fall very short in an area where most semantic representations have felt some obgligation to succeed, namely intepretation and inference. That AMR has made this compromise is no oversight, but an informed design choice. Version 1.2 of the AMR specication states that by default an AMR does not indicate how it should be processed (Banarescu et al., 2012). Indeed, besides concepts that can be interpreted as propositional connectives like `:and`, `:or`, and `:cond`, there is little that can be directly inferred from a basic AMR besides possibly lexical inference on the concepts and relations. Performing lexical inference directly would require at least extensive hand authoring of relationships between PropBank frames, and in general is beyond the scope of this paper.

Finding ways to support inference for AMR is no trivial curiousity. For machine-based natural language understanding, it is arguable that a semantic representation needs to deeply reflect natural language at the expressive level while supporting complex reasoning and capturing certain aspects of knowledge (Schubert, 2015). Furthermore, inferential methods are critical to being able to guarantee, verify, and discover properties of software systems. One relevant application is discovering and resolving bias in NLP systems (Gordon and Durme, 2013). AMR extensions ameliorate the issue of expressivity by integrating more semantic information into an AMR while still retaining its minimal and predicative structure. Pustejovsky et al. (2019) makes direct inference on AMR significantly easier by providing an unambiguous way to interpret an AMR that makes use of quantification, negation, or modality, while retaining the predicative core of AMR. Donatelli et al. (2018) extends AMR with a treatment of tense and aspect which allows for direct inference involving temporal information. We will show that integrating these two AMR extensions allows for interpretation into a highly expressive logic we call first-order hybrid tense logic *FHTL*, which merges hybrid logic equipped with first order features like (actualist) quantification (Hansen, 2007), and hybrid logic equipped for reasoning about time and tense (Blackburn and Jørgensen, 2012). We present a sound and complete tableau-based proof procedure for *FTHL* sentences. The tableau method is a well known approach to theorem proving that has been adapted to many logics, it works by extracting a tree from a sentence in a logic, where the nodes consist of root subformulae or supporting formulae to be proved or refuted. We also provide a modified version of this tableau method, which we use to check the satisfiability of translated extended AMR sentences in a finite model, and a method for translating an extended AMR into a *FHTL* formula.

2 Related Work

2.1 Semantic Features in AMR and Possbility of Inference

In recent years there have been more efforts towards extension and interpretation of AMR in ways more suitable to direct inference. Bos (2016) provides a means of translating standard AMR to first-order predicate logic with negation, and that an AMR without recurrent variables under this translation is in the decidable two-variable fragment of first-order predicate logic. (Donatelli et al., 2018) extends sentential AMR to incorporate a coarse grained treatment of tense and aspect, both of which affords inference, and in a very rich way in the case of aspect, as we will discuss at the end. O'Gorman et al. (2018) describes a multi-sentential treatment of AMR with a corpora of examples, for handing document level semantics in addition to AMR's current sentential semantics. This has possible implications for document-level and discourse-level inference in AMR, à la discourse representation theory. Pustejovsky et al. (2019) and its extension of AMR with quantification, scope, negation, and modality, is effectively a royal road to interpretation of AMR into first-order logic or related variants, and we will frequent use of it in section 5. (Lai et al., 2020) provides a continuation-based semantics for translating AMR into first-order predicate logic in a way that preserves projective phenomena. While this influenced our interpretation approach and its translation method is much more elegant than ours, we elected not to use it since it uses a Neo-Davidsonian representation for the target first-order predicate logic semantics, which as we will is more expensive to handle via tableau-based reasoning since it guarantees that there is at least one quantifier in the interpretation for every relation symbol. Finally, Bonial et al. (2020) provides an AMR extension designed for faciliatating natural language understanding in dialogue systems integrating both tense and aspect and a speech act inventory, which conveys the illocutionary force of a statement accompanying the content. This is another extension that affords the possbility of discourse level inference in addition to sentential inference, which while offering rich conclusions will require more a sophisticated semantics sensitive to context.

2.2 Hybrid Logic and Its Extensions

Hybrid logic is an extension of the propositional modal logic K (K has no conditions on the modal frame of its underlying models) allowing for explicit reference to modal states/worlds through a prefix notation. Where ordinary modal logic writes $\Diamond p$ to indicate that there is some world where p holds at some world accessible from that world, hybrid logic by default writes one of $a\Diamond p$, $a{:}\Diamond p$, or $@_a\Diamond p$ (we use the latter notation from here on), to indicate p holds at some world accessible from a specifically. a in this notation is a *nominal*, which uniquely names a world in the underlying model. A *world* in a modal or hybrid model for us is essentially just a maximal set of sentences in the language, that is for any proposition p and nominal a we have either $@_a p$ or $@_a\neg p$. Worlds are used to model any notion consistent with it, usually possible states of affairs, in particular we will use them to describe states of affairs at different points in time.

For propositional hybrid logic, we have a set of propositional symbols $\mathsf{PROP} = \{p, q, r, \ldots\}$, modalities $\mathsf{MOD} = \{m, m', m'', \ldots\}$, and *nominals* $\mathsf{NOM} = \{i, j, k, \ldots\}$, all nonempty and disjoint from each other. Nominals name states of a model of the logic, usually these states are possible worlds. The basic hybrid language over PROP, MOD, and NOM is given by the grammar:

$$\varphi ::= i \mid p \mid \neg\psi \mid (\psi_1 \wedge \psi_2) \mid (\psi_1 \vee \psi_2) \mid (\psi_1 \rightarrow \psi_2) \mid \langle \mathrm{M}\rangle\psi \mid [\mathrm{M}]\psi \mid @_i\varphi$$

Where we have only one modality m we write $\Diamond$ and $\Box$ for $\langle m\rangle$ and $[m]$. For instance, where the propositional symbol r denotes *Rabia owns a car*, the following are (equivalent) theorems of hybrid logic:

- *It is not possible for Rabia to own a car and for Rabia not to own a car.*

$$\neg\Diamond(r \wedge \neg r)$$

- *Necessarily, Rabia owns a car or does not own a car.*

$$\Box(r \vee \neg r)$$

We can use nominals to add the notion of time to these sentences, for instance where i denotes the world at 2015 and j denotes the world at 2021, if we want to indicate that Rabia owned a car in 2015, and that whatever was the case at 2015 is possible in 2021, we can write: $@_i r \wedge @_j \Diamond i$

2.2.1 First-order Hybrid Logic

First-order modal logic (or quantified modal logic / *QML*) extends first-order predicate logic in a way analogous to how propositional modal logic extends first order propositional logic. An introduction to both QML and propositional modal logic can be found in Fitting and Mendelsohn (1998). As before, a tableau method is a proof procedure for some first-order language (and a decision procedure for a propositional language), and it is the most widely used proof procedure for modal logics (Girle, 2014). In a sense, hybrid logic simply makes explicit through the satisfaction operator what is implicit in modal logic, and indeed, in modal tableau prefixes play an explicit role. As a result the tableau method generalizes to hybrid logics very well. Blackburn and Marx (2002) provides the first tableau method for a first-order extension of hyrbid logic. Quantified hybrid logic *QHL* as defined in Blackburn and Marx (2002) unfortunately for us, uses constant-domain semantics and possibilist quantification (where quantifiers range over all the collective domain of all worlds/times), making it inconvenient for modeling quantification and reference in natural language. Furthermore it omits non-unary function symbols, which are a necessity for modeling AMR concepts as `have-rel-role` in first-order language. Hansen (2007) remedies this by developing a semantics and tableau method for first-order hybrid logic *FHL*, a variant of *QHL* using varying-domain semantics and actualist quantification (quantifiers only range over entities at the time of reference). As can be inferred from the name, *FHL* is the core component of our development of *FHTL* in section 3.

2.2.2 Basic Hybrid Tense Logic

The roots of hyrbid logic, and hybrid tense logic in particular, range back to the philosopher Arthur N. Prior's work in the 1960s (Prior and Hasle, 2003). We take our treatment of basic hybrid tense logic *BHTL* from Blackburn and Jørgensen (2012), which roughly can be seen as renaming $\Diamond$ and $\Box$ to F and G in basic hybrid logic (the modality for the future), and adding their duals under the accessibility relation P and H (the modality for the past), sometimes written $\Diamond^{-1}$ and $\Box^{-1}$ For intuition, if we have $@_i F\varphi$ for some time i, then we have that φ holds at some time in the future from the perspective of i. If we have $@_i G\varphi$, then φ holds at every world in the future from the perspective of i. Similarly for $@_i P\varphi$ and $@_i H\varphi$, except both statements refer to the past from the perspective of i. Now that the accessibility relation R of an interpretation of *BHTL* notionally models time, there additional properties of R to consider (such as it being a strict partial order) when proving soundness and completeness for the language. This is accomplished in Blackburn and Jørgensen (2012), but in this paper we avoid these considerations and make no assumptions about the model's accessibility relation. Basic hybrid tense logic can be used to model tense at the sentential and discourse leves. For the sentential level, where again r denotes *Rabia owns a car*

- *That Rabia owns a car now, implies that Rabia will own a car.*

$$@_n(r \to F(r))$$

  (This is a theorem only if we make certain assumptions about time.)

For the discourse level, consider the sentences *John entered the room. It was dark. He sat down.* We can represent them as:

$$P(i \wedge \textit{john-enter-room}) \wedge P(j \wedge \textit{room-is-dark}) \wedge @_i j \wedge P(k \wedge \textit{john-sits-down}) \wedge @_k Pi$$

The event of John entering the room is associated with the state i, and the room being dark is associated with the state j. We have $@_i j$ to indicate that they occur at the same time, $@_k Pi$ to indicate that they happened in the past of the state k associated with John sitting down.

3 First-order Hybrid Tense Logic

Here we give a modification of Hansen's First-order Hybrid Logic which describes tense instead of modality while retaining *FHL*'s varying domain semantics and actualist quantification.

3.1 Syntax of *FHTL*

The syntax of *FHTL* is identical to *FHL* as given in Hansen (2007), except $\Box$ and $\Diamond$ are replaced by their semantic equivalents F and G, and their dual temporal modalities P and H are added. The *FHTL* language contains the following (countably infinite) sets: a set of first-order variables FVAR, a a set of constants CON, a set of relation symbols RSYM, a set of function symbols FSYM, a set of nominals NOM, a set of state variables SVAR.

Definition 3.1 (*FHTL*–terms)**.** Terms in *FHTL* are generated by the following grammar:

$$t ::= x \mid c \mid u{:}t \mid f(t_1, \ldots, t_n)$$

Where $x \in \mathsf{FVAR}$, $c \in \mathsf{CON}$, $u \in \mathsf{NOM} \cup \mathsf{SVAR}$, and f is an n–ary function symbol in FSYM.

Definition 3.2 (*FHTL*–formulae)**.** *FHTL*–formulae are generated from the atomic formulae according to the following rules:

$$\varphi ::= R(t_1, \ldots, t_n) \mid t_1 = t_2 \mid u \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid (\exists x)\psi \mid F\psi \mid P\psi \mid @_n\psi$$

Where $R \in \mathsf{RSYM}$, $t_1, \ldots, t_n$ are *FHTL* terms, $u \in \mathsf{NOM} \cup \mathsf{SVAR}$, and $x \in \mathsf{FVAR}$. We omit definitions for $\wedge$, $\rightarrow$, H, G, and $\forall$ for the sake of simplicity (and space), since they can be defined in terms of the other rules.

For example, we can represent the following tense and quantification sensitive natural language sentences as follows:

- *There currently is a person who will be president.*

$$@_{\mathit{now}}(\exists x)(\mathit{Person}(x) \wedge F(\mathit{President}(x)))$$

- *For every person, they have a parent who existed before them.*

$$(\forall x)(\mathit{Person}(x) \rightarrow P((\exists x)\mathit{Parent}(x, y)))$$

3.2 Semantics of *FHTL*

Since we want the domain of quantification to be indexed over the collection of nominals/times, we look to Fitting and Mendelsohn's (1998) treatment of first-order modal logic with varying domain semantics and use it to alter the *FHL* model definition to the following:

$$(T, \mathcal{R}, (D_t)_{t \in T}, I_{nom}, (I_t)_{t \in T})$$

Thus with varying domain semantics a *FHTL* model is identical to the definition for a *FHL* model in that:

- (T, R) is a modal frame.
- I_{nom} is a function assigning members of T to nominals.

The differences manifest with regard to the model and interpretation. Namely, where $D = \cup_{t \in T} D_t$, (D, I_t) is a first-order model where:

- $I_t(q) \in D$ where q is a unary function symbol[1].
- $I_t(R) \in D^k$ where R is a k-ary relation symbol.

If we were using constant domain semantics we would require $I_t(c) = I_{t'}(c)$ for any constant c and any times $t, t' \in T$. But we do not require this since the interpretation of the constant need not exist at both times. This permits us to distinguish between the domain of a frame (everything that exists in the "history" of the model) and the domain of a time, in a way that prevents any first-order variable x from failing to refer at a given time, even if it has no interpretation at that time. Intuitively this permits *FHTL* to handle interpretation of entities in natural language utterances, which while reasonable to refer to, do not exist at a current time, e.g. 19th century US presidents and unknown future US presidents.

Free variables are handled similarly as in *FHL*. Where again $D = \cup_{t \in T} D_t$, a *FHTL* assignment is a function:

$$g{:}\mathsf{SVAR} \cup \mathsf{FVAR} \rightarrow T \cup D$$

Where state variables are sent to times/worlds and first-order variables are sent to D, the domain of the frame. Thus given a model and a variable assignment g, the interpretation a term t denoted by $\bar{t}$ is defined by:

- $\overline{x} = g(x)$ for x a variable and any $t \in T$.
- $\overline{c} = I_t(c)$ for c a constant and some $t \in T$.

[1]We use these for non-rigid designation and constants for rigid designation, e.g. roles versus names.

- $\overline{f(t_1,...,t_n)} = I_w(f)(\overline{t_1},...,\overline{t_n})$ for f an n-ary function symbol, $t_1,\ldots,t_n$ terms, and $w \in T$.
- $\overline{i{:}t} = I_w(i{:}t) = I_{I_{nom}(i)}(t)$ for t a term, i a nominal, and $w \in T$.
- $\overline{u{:}t} = I_w(u{:}t) = I_{g(u)}(t)$ for t a term, u a state variable, $w \in T$.

Finally we say an assignment g' is an x-variant of g if g' and g agree on all variables except possiblly x. In particular, we say g' is an x-variant of g at a time t, if g' and g on all variables except possiblly x and $g'(x) \in D_t$. Finally, given a model $\mathfrak{M}$, a variable assignment g, and a state s, we have the inductive definition of $\mathfrak{M}, s \vDash_g \varphi$ as:

$$\begin{array}{ll}
\mathfrak{M}, s \vDash_g R(t_1,\ldots,t_n) & \iff \langle \overline{t_1},\ldots,\overline{t_n}\rangle \in I_s(R) \\
\mathfrak{M}, s \vDash_g t_i = t_j & \iff \overline{t_i} = \overline{t_j} \\
\mathfrak{M}, s \vDash_g n & \iff I_{nom}(n) = s, \text{for } n \text{ a nominal} \\
\mathfrak{M}, s \vDash_g w & \iff g(w) = s, \text{for } w \text{ a state variable} \\
\mathfrak{M}, s \vDash_g \neg\varphi & \iff \mathfrak{M}, s \nvDash_g \varphi \\
\mathfrak{M}, s \vDash_g \varphi \vee \psi & \iff \mathfrak{M}, s \vDash_g \varphi \text{ or } \mathfrak{M}, s \vDash_g \psi \\
\mathfrak{M}, s \vDash_g (\exists x)\varphi & \iff \mathfrak{M}, s \vDash_{g'} \varphi \text{ for some } x\text{-variant } g' \text{ of } g \text{ at } s \\
\mathfrak{M}, s \vDash_g F\varphi & \iff \mathfrak{M}, t \vDash_g \varphi \text{ for some } t \in T \text{ such that } \mathcal{R}st \\
\mathfrak{M}, s \vDash_g P\varphi & \iff \mathfrak{M}, t \vDash_g \varphi \text{ for some } t \in T \text{ such that } \mathcal{R}ts \\
\mathfrak{M}, s \vDash_g @_n\varphi & \iff \mathfrak{M}, I_{nom}(n) \vDash_g \varphi \text{ for } n \text{ a nominal} \\
\mathfrak{M}, s \vDash_g @_w\varphi & \iff \mathfrak{M}, g(w) \vDash_g \varphi \text{ for } w \text{ a state variable}
\end{array}$$

As usual, a formula is satisfiable if there is some model/interpretation which makes the formula true, and a formula is valid if every model/interpretation which makes the formula true. We also note that if ψ is a sentence (a formula with no free variables, also called a closed formula), then whether $\mathfrak{M}, s \vDash_g \varphi$ does not depend on the variable assignment g.

3.3 The Tableau Calculus

When working with the tableau calculus, following Fitting and Mendelsohn (1998) we assume for each nominal or state variable s, there is a countably infinite list of parameters PAR, where parameters are free variables which are never quantified over, arranged in such a way that different nominals/state variables never share the same parameter. We write $s{:}p$ to indicate a parameter is associated with a nominal/state variable s, just as we do with terms. The *extended language* is the language obtained by adding PAR to the standard *FHTL* language. In the tableau rules, a term in the extended language is called a closed term if it contains no first-order variables or state variables.

3.4 Soundness and completeness

The proof of soundness for *FHTL* follows from the soundness of *FHL* established in Hansen (2007), with additions for the rules (P), $(\neg P)$, **(P)–bridge**, and **(P/F)–trans**, the soundness of which is easy to demonstrate. The proof of completeness is essentially the same as the one given in Bolander and Braüner (2006) for Blackburn's tableau system for standard hybrid logic ($\mathcal{HL}(@)$) with accomodations made for the first order aspects of *FHTL* adapted from Hansen (2007). While Bolander and Braüner (2006) proves termination for Blackburn's tableau system, we cannot do that for *FHTL*, since a terminating tableau system would be equivalent to a general decision procedure for first-order predicate logic. We will however intoduce restrictions on the quantifier and term rules in the next section which will lead to terminating tableau constructions at the cost of general completeness, which will be no real loss since the purpose of our revised tableau system is for finite model checking.

Remark 3.1. From here on we normalize formulae such that a negation symbol only occurs within the scope of the satisfaction operator, that is we will always write $@_s\neg\varphi$ and never $\neg @_s\varphi$ from now on, even though the two are semantically equivalent.

Definition 3.3 (Closed and open). If a tableau branch contains a formula $@_s\varphi$ and its negation $@_s\neg\varphi$ we say the branch is *closed*. If every branch of the tableau is closed we say the tableau itself is closed. If a tableau or branch is not closed we say it is *open*.

Propositional rules:

$$\frac{@_s(\varphi \lor \psi)}{@_s\varphi \mid @_s\psi}\,(\lor) \qquad \frac{@_s\neg(\varphi \lor \psi)}{\begin{array}{c}@_s\neg\varphi\\ @_s\neg\psi\end{array}}\,(\neg\lor) \qquad \frac{@_s\neg\neg\varphi}{@_s\varphi}\,(\neg\neg)$$

Modal rules:

$$\frac{@_sF\varphi}{\begin{array}{c}@_sFa\\ @_a\varphi\end{array}}\,(F)^{ab} \qquad \frac{@_sP\varphi}{\begin{array}{c}@_sPa\\ @_a\varphi\end{array}}\,(P)^{ab} \qquad \frac{@_s\neg P\varphi \qquad @_sPt}{@_t\neg\varphi}\,(\neg P) \qquad \frac{@_s\neg F\varphi \qquad @_sFt}{@_t\neg\varphi}\,(\neg F)$$

Quantifier rules:

$$\frac{@_s(\exists x)\varphi}{@_s\varphi[s{:}p/x]}\,(\exists)^{c} \qquad \frac{@_s\neg(\exists x)\varphi}{@_s\neg\varphi[t/x]}\,(\neg\exists)^{d}$$

@ rules:

$$\frac{@_s@_t\varphi}{@_t\varphi}\,(@) \qquad \frac{@_s\neg@_t\varphi}{@_t\neg\varphi}\,(\neg@)$$

$$\frac{@_st \qquad @_s\varphi}{@_t\varphi}\,\textbf{(nom)} \qquad \frac{[i \text{ on the branch}]}{@_ii}\,\textbf{(nom ref)}$$

$$\frac{@_sPt}{@_tFs}\,(P\textbf{–trans}) \qquad \frac{@_sFt}{@_tPs}\,(F\textbf{–trans})$$

$$\frac{@_sPt \qquad @_tu}{@_sPu}\,(P\textbf{–bridge}) \qquad \frac{@_sFt \qquad @_tu}{@_sFu}\,(F\textbf{–bridge})$$

[a]The nominal a is new to the branch.
[b]The formula φ is not a nominal.
[c]$s{:}p$ is new to the branch.
[d]p is any parameter which exists at s.

A closed tableau is a proof of the unsatisfiability of the tableau's root formula, i.e. there is no model or assignment of variables in which it holds. Thus a tableau proof of an *FHTL* sentence φ is a closed tableau with the root $@_s\neg\varphi$ for some nominal s not occuring in φ.

Definition 3.4 (Accessibility Formula). A formula occurence in a tableau is called an accessibility formula if it is of the form $@_aFb$ or $@_aPb$ and is the (first) conclusion of an application of the rule F or P respectively.

Definition 3.5 (Quasi-subformula). A formula φ is a *quasi-subformula* of a formula ψ if one of the the following is the case:

1. φ is a subformula of ψ modulo substitution of free variables in φ for parameters, and full or partial substitution of a term with an equivalent term.
2. φ is of the form $\neg\chi$ where χ is a subformula of ψ modulo substitution of free variables in χ for parameters, and full or partial substitution of a term with an equivalent term.

Equality rules:

$$\frac{}{@_i j{:}t = j{:}t}\ \textbf{(ref)}^a \qquad \frac{@_i j{:}t = k{:}s \qquad @_i\varphi}{@_i\varphi[j{:}t//k{:}s]}\ \textbf{(sub)}^b$$

***FHTL* term rules:**

$$\frac{@_i k_1{:}t = k_2{:}s}{@_i k_1{:}t = k_2{:}s}\ \textbf{(:1)} \qquad \frac{@_i j}{@_k i{:}t = j{:}t}\ \textbf{(:2)}^a \qquad \frac{}{@_i k{:}j{:}t = j{:}t}\ \textbf{(:3)}^a$$

$$\frac{@_i R(t_1,...,t_n)}{@_i R(i{:}t_1,...,i{:}t_n)}\ \textbf{(:fix 1)} \qquad \frac{@_i \neg R(t_1,...,t_n)}{@_i \neg R(i{:}t_1,...,i{:}t_n)}\ \textbf{(:fix 2)} \qquad \frac{@_i t = s}{@_i i{:}t = i{:}s}\ \textbf{(:fix 3)} \qquad \frac{@_i \neg t = s}{@_i \neg i{:}t = i{:}s}\ \textbf{(:fix 4)}$$

$$\frac{}{@_i f(t_1,...,t_n) = f(i{:}t_1,...,i{:}t_n)}\ \textbf{(:func)}^c$$

[a]Where t is a closed term.
[b]$\varphi[j{:}t//k{:}s]$ is φ where some occurences of $j{:}t$ have been replaced by $k{:}s$.
[c]f is an n–ary function symbol and $t_1,...,t_n$ are closed terms.

Altering the definition to allow parameters being substituted for free variables, and full or partial substitution of any term with an equivalent term ensures compatibility of the following proofs with the quantifier rules, term rules, and equality rules. We say a formula $@_s\varphi$ on a *FHTL* tableau branch Θ is a *root subformula* if φ is a quasi subformula of the root formula of the tableau.

Remark 3.2. If $@_s\varphi$ occurs on a branch Θ we may write $@_s\varphi \in \Theta$.

Lemma 3.1 (**Root Subformula Property**). *If a formula $@_s\varphi$ occurs in a FHTL tableau where φ is not of the form a, Fa, or Pa for a a nominal, or $u = t$ for optionally prefixed closed terms t and u, then φ is a (positively occuring) root subformula. If $@_s\neg\varphi$ occurs in a FHTL tableau then φ is a root subformula, but we say it is negatively occuring.*

Proof. This is verified by checking the tableau rules with attention to the fact that if the conclusion of a rule is of the form $@_s\neg\psi$ then its premise is of the form $@_s\neg\varphi$. □

Definition 3.6. An occurence of a nominal in a formula is *equational* if the occurence is a formula (i.e. not part of a satisfaction operator).

For instance, the occurence of a nominal c is equational in the formula $@_a(c \vee (\exists x)\psi)$ but not in the formula $@_c(\exists x)\psi$.

Definition 3.7 (N_Θ)**.** The set of nominals which occur on a branch Θ is written N_Θ

Definition 3.8 ($\sim_\Theta$)**.** Where Θ is a *FHTL* tableau branch, define a binary relation $\sim_\Theta$ on N_Θ by $a \sim_\Theta b$ if and only if $@_a b$ occurs on Θ. Let $\sim^*_\Theta$ be reflexive, transitive, and symmetric closure of $\sim_\Theta$.

Theorem 3.2. *Where $@_a b$ occurs on a branch Θ, if the nominals a and b are distinct, then each has the property that it is identical to, or related by $\sim_\Theta$ to, a nominal with a positive and equational occurence in the root subformula.*

Proof. This is verified by checking each rule, in some cases making use of 3.1. **(nom ref)** might initially appear to be a problem, since we are permitted to apply it for any nominal on the branch, but the nominals in its conclusions are not distinct. For F and P we make use of the restriction that they cannot be applied to premises of the form $@_a(F/P)\varphi$ where φ is a nominal. □

Theorem 3.3. *Let $@_aFb$ be a formula occurence on a branch Θ of a tableau. Either there is a positively occuring subformula Fb' of the root formula such that $b \sim^*_\Theta b'$ or there is an accessibility formula occurence $@_{a'}Fb'$ such that $a \sim^*_\Theta a'$ and $b \sim^*_\Theta b'$. Similarly for a formula occurence $@_aPb$.*

Proof. This is verified by checking each of the tableau rules, in some cases making use of 3.1. $\square$

Definition 3.9 ($\subseteq_\Theta$)**.** Where a and b are nominals occuring on an *FHTL* tableau branch Θ, a is *included* in b with respect to Θ if for any root subformula φ, if $@_a\varphi$ occurs on Θ then $@_b\varphi$ also occurs on Θ, similarly for their negations. If a is included in b with respect to Θ, and the first occurence of b on Θ is before the first occurence of a on Θ, then we write $a \subseteq_\Theta b$.

Definition 3.10 ($W, \approx$)**.** Let W be the subset of N_Θ containing any nominal a with the property that there is no nominal b such that $a \subseteq_\Theta b$. Let $\approx$ be the restriction of $\sim_\Theta$ to W.

We can see that W contains every nominal in the root formula, and since any branch Θ is closed under **(nom ref)** and **(nom)** we have that $\sim_\Theta$ and $\approx$ are equivalence relations. For a nominal a in W we write $[a]$ to denote the equivalence class of a and $W/_\approx$ to denote the set of equivalence classes.

Definition 3.11. Let R be the binary relation defined on W defined by Rac if and only if there are nominals $a' \approx a$ and $c' \approx c$ satisfying one of the following conditions:

1. The formula $@_{a'}Fc'$ or $@_{a'}Pc'$ occurs at Θ and was introduced to the branch by F or P respectively.
2. There is a nominal $d \in N_\Theta$ such that the formula $@_{a'}Fd$ or $@_{a'}Pd$ was introduced to the branch by F or P respectively and $d \subseteq_\Theta c'$
3. The formula $@_{a'}Fc'$ or $@_{a'}Pc'$ occurs at Θ and a' or c' occurs in the root formula.

Note that the nominal d in the second item is not an element of W and that the accessibility relation R is compatible with $\approx$. We define $\overline{R}$ as the binary relation on $W/_\approx$ defined by $\overline{R}[a][c]$ if and only if Rac.

$(W, \overline{R})$ are a modal frame, the first step towards constructing a model of an open tableau branch Θ.

3.5 First-Order Components for Completeness

Completeness for a tableau system is demonstrated by giving a method for constructing a model for an open branch such that it satisfies every formula on the branch. Taking our cue from Hansen (2007) and making accomodations for the tableau rules and and structure we adapted from Bolander and Braüner (2006), we construct the first-order components of our model. Here we define the analogues of $D_{[w]}, I_{[w]}$ for $[w] \in W/_\approx$ and I_{nom}. To start we let $\mathcal{D}$ be set defined by:

$$\mathcal{D} = \{i{:}t \mid \text{for some } i \in W \text{ and some closed term of the extended language } t\}$$

Further we define a relation $\equiv$ on D by: $i{:}t \equiv j{:}s \iff @_k i{:}t \equiv j{:}s \in \Theta$ for some $k \in W$ From the tableau rules **(ref)**, **(sub)**, and **(:1)**, we observe $\equiv$ is an equivalence relation on $\mathcal{D}$. From here we define the domain D of the model as $\mathcal{D}/_\equiv$, with the elements written as $\overline{i{:}t}$, for all $[w] \in W/_\approx$ define:

$$D_{[w]} = \{\overline{j{:}p} \mid j \in [w] \text{ and } p \text{ a parameter}\}$$

For a a constant or parameter and $[w] \in W/_\approx$ we define $I_{[w]}(a) = \overline{w{:}a}$ which is well defined by **(:2)**. For an n-ary relation symbol R and $[w] \in W/_\approx$ define $I_{[w]}(R)$ by:

$$I_{[w]}(R)(\overline{i_1{:}t_1}, ..., \overline{i_n{:}t_n}) \iff @_w R(i_1{:}t_1, ..., i_n{:}t_n) \in \Theta$$

for all $\overline{i_1{:}t_1}, ..., \overline{i_n{:}t_n} \in D$. The interpretation is similar for n-ary function symbols. And naturally for interpretation of nominals, for $w \in W$ define $I_{nom}(w) = [w]$.

Before finally demonstrating that this model $\mathfrak{M}$ realizes every formula on the tableau branch from which it was constructed we need the following lemma.

Lemma 3.4. *If t is a closed term of the extended language and i is a nominal then we have $I_{[i]}(t) = \overline{i{:}t}$ for all variable assignments g in our model $\mathfrak{M}$.*

Proof. The proof is by induction on the construction of t. By our assumption t cannot a first-order variable, and if t is a constant or a parameter then

$$I_{[i]}(t) = \overline{i{:}t}$$

by definition. Suppose alternatively t is of the form $u{:}s$ for $u \in W$, since t contains no variables neither does s and consequently:

$$I_{[i]}(u{:}s) = I_{[u]}(s) \overset{*}{=} \overline{u{:}s} \overset{\textbf{(:3)}}{=} \overline{i{:}u{:}s}$$

Where $*$ follows from the inductive hypothesis and **(:3)** from **(:3)**. Similarly for the case involving functional symbols. $\square$

Lemma 3.4 is made use of in the cases of formulae which openly depend on terms in the completeness proof.

Theorem 3.5 (**Completeness of the** *FHTL* **Tableau Method**). *Assuming the branch* Θ *is open, for any formula* $@_a\varphi$ *which contains only nominals from* W*, the following two statements hold:*

$$@_a\varphi \in \Theta \implies \mathfrak{M}, [a] \vDash_g \varphi \textit{ for some (all) variable assignments } g$$
$$@_a\neg\varphi \in \Theta \implies \mathfrak{M}, [a] \nvDash_g \varphi \textit{ for some (all) variable assignments } g$$

Proof. This is demonstrated by induction on the structure of the formula. We present the most difficult case, namely where φ is of the form $F\psi$.

Assume $@_aF\psi \in \Theta$. Then we need to show that there is an equivalence class $[c] \in W/_\approx$ such that $R[a][c]$ and $\mathfrak{M}, [c] \vDash_g \psi$ for some variable assignments g. We have two cases based on the question of whether or not ψ is a nominal. In the case where

- ψ *a nominal* Suppose ψ is some nominal b, we only need prove $\overline{R}[a][b]$. By theorem 3.3 there is either a root nominal b' such that $b' \sim_\Theta b$ or there is an accessibility formula $@_{a'}Fb' \in \Theta$ such that $a' \sim_\Theta a$ and $b' \sim_\Theta b$. In the first case then we have $@_aFb' \in \Theta$ and $b' in W$ so $\overline{R}[a][b']$ and since $b' \approx b$ (since both are in W) we have $[b'] = [b]$ and we are done. In the second case, there are four subcases depending on whether or not a' and a are equal and b' and b are equal:
  - $a' = a$ and $b' = b$:
    Then we have $\overline{R}[a][b]$ and are done.
  - $a' = a$ and $b' \neq b$:
    By 3.2 there is a nominal c occuring in the root formula such that $b' \sim_\Theta c$. But then $@_aFc \in \Theta$ and $c \in W$ so $\overline{R}[a][c]$, and by the definition of $\approx$ we have $[c] = [b]$ and are done.
  - $a' \neq a$ and $b' = b$:
    By 3.2 there is a nominal c occuring in the root formula such that $a' \sim_\Theta c$. But then $@_cFb \in \Theta$ and $c \in W$ so $\overline{R}[c][b]$, and by the definition of $\approx$ we have $[c] = [a]$ and are done.
  - $a' \neq a$ and $b' \neq b$:
    By 3.2 there are nominals c and d occuring in the root formula such that $a' \sim_\Theta c$ and $b' \sim_\Theta d$. But then also $@_cFd \in \Theta$, and $c, d \in W$ so $\overline{R}[c][d]$ and by definition of $\approx$ we have $[c] = [a]$ and $[d] = [b]$ and are done.
- ψ *not a nominal* Since $a \in W$ the application of (F) to $@_a\psi$ is not blocked by the loop-check condition of the tableau construction algorithm. From the rule (F), there are formulae $@_aFc, @_c\psi \in \Theta$ where the nominal c is new to the branch. If $c \in W$ then $\overline{R}[a][c]$ and by induction $\mathfrak{M}, [c] \vDash_g \psi$ for some variable assignments g. If $c \notin W$ then by the definition of W there is a nominal d where $c \subseteq_\Theta d$. Without loss of generality assume there is no nominal e such that $d \subseteq_\Theta e$, this implies $d \in W$. Additionally by 3.1 ψ is a root subformula and we have $@_d\psi \in \Theta$. By induction we have $\mathfrak{M}, [d] \vDash_g \varphi$, and by the definition of W and $\approx$ we have $\overline{R}[a][d]$.

Now assume $@_a\neg F\psi \in \Theta$, we need to show $\mathfrak{M}, [a] \nvDash_g F\psi$ does not hold for any variable assignment g, i.e. for any $[c] \in W/_\approx$ where $\overline{R}[a][c]$ that $\mathfrak{M}, [c] \nvDash_g \psi$ for any variable assignment g. From $\overline{R}[a][c]$ we have that there are nominals $a' \approx a$ and $c' \approx c$ satisfying one of the three conditions of R in definition 3.11. In the first and third conditions of the definition we have $@_{a'}Fc' \in \Theta$ and as a result $@_aFc \in \Theta$ and in turn by ($\neg F$) $@_c\neg\psi \in \Theta$. By induction we have $\mathfrak{M}, [c] \nvDash_g \psi$ for every variable assignment g. In the second condition of R in definition 3.11 there is a nominal $d \in N_\Theta$ such that $@_{a'}Fd \in \Theta$ and $d \subseteq_\Theta c'$, consequently $@_aFd \in \Theta$ and by ($\neg F$) $@_d\neg\psi \in \Theta$. But by lemma 3.1 ψ is a root subformula, and since $d \subseteq_\Theta c'$ we have $@_{c'}\neg\psi \in \Theta$. By induction we have $\mathfrak{M}, [c'] \nvDash_g \psi$ for every variable assignment g and by the definition of $\approx$ also $[c'] = [c]$. □

4 *FHTL* Model Checking with Finite Tableaux

For our task of AMR inference, we are not always concered with the determining the general satisfiability or validity of an AMR formula translated into *FHTL*, but rather whether it holds in a model consistent with an established set of *FTHL* translations of AMR sentences.

This is because reasoning about AMR sentences (or eventually even about AMR representing discourses or documents) takes place in a particular context or a general kind of context. These contexts are described in logic by models, and where they are determined by some finite set of AMR sentences, these models will necessarily be finite, since across any finite number of AMR sentences only a finite number of times (nominals), entities (terms of the extended language), and relationships between entities (function symbols and relation symbols) can be referenced. It is possible for some contexts to require non-finite models, or for some model conditions to be so general as to require automated theorem proving techniques for assessing a sentence rather than automated model checking, but these issues are beyond the scope of this paper.

For *FHTL* sentences and finite models, we have an instance of a local model-checking problem where given formula φ, a finite *FHTL* model structure $\mathfrak{M}$, a time t in $\mathfrak{M}$, and a variable assignment g, we need to determine whether $\mathfrak{M}, t \models_g \varphi$ (Müller-Olm et al., 1999).

Here we develop a restriction of our method for tableau construction, which ensures termination of tableau construction (i.e. all resulting tableaux are finite) at the cost of completeness. This is actually an easy compromise for us to make. Model checking via tableau does not benefit from the completeness of a tableau method, since we are not trying to demonstrate a sentence's validity or unsatisfiability, but are interested only whether it holds in the relevant model. Tableau termination however is critical to decidability, and since we are assuming finite models, our approach had better be decidable! Our approach to using terminating tableaux as a means of model checking for *FHTL* based on the approach in Bohn et al. (1998), and makes use of the idea of quantifier depth from Fitting (1988).

4.1 Restricted rules

Equality rules:

$$\frac{[i \text{ and } j \text{ on the branch.}]}{@_i j{:}t = j{:}t}\ \textbf{(ref)}^{a} \qquad \frac{@_i j{:}t = k{:}s \qquad @_i\varphi}{@_i\varphi[j{:}t//k{:}s]}\ \textbf{(sub)}^{b}$$

***FHTL* term rules:**

$$\frac{@_i j_1{:}t = j_2{:}s}{@_k j_1{:}t = j_2{:}s}\ \textbf{(:1)}^{c} \qquad \frac{@_i j}{@_k i{:}t = j{:}t}\ \textbf{(:2)}^{cd} \qquad \frac{}{@_k j{:}i{:}t = i{:}t}\ \textbf{(:3)}^{cd}$$

$$\frac{@_i R(t_1, ..., t_n)}{@_i R(i{:}t_1, ..., i{:}t_n)}\ \textbf{(:fix 1)}^{e} \qquad \frac{@_i \neg R(t_1, ..., t_n)}{@_i \neg R(i{:}t_1, ..., i{:}t_n)}\ \textbf{(:fix 2)}^{e}$$

$$\frac{@_i t = s}{@_i i{:}t = i{:}s}\ \textbf{(:fix 3)}^{e} \qquad \frac{@_i \neg t = s}{@_i \neg i{:}t = i{:}s}\ \textbf{(:fix 4)}^{e}$$

$$\frac{[i \text{ and one of the two function terms on the branch}]}{@_i f(t_1, ..., t_n) = f(i{:}t_1, ..., i{:}t_n)}\ \textbf{(:func)}^{f}$$

[a] t is a closed term on the branch, j not among its prefixes.
[b] $\varphi[j{:}t//k{:}s]$ is φ where some occurences of $j{:}t$ have been replaced by $k{:}s$.
[c] k on the branch.
[d] t is a closed term on the branch, and i, j are distinct nominals on the branch, not among the prefixes of t.
[e] i is not among the prefixes of the terms in the conclusion.
[f] f is an n–ary function symbol and $t_1, ..., t_n$ are closed terms with i not among their prefixes.

Definition 4.1 (Saturation). A tableau branch is *saturated* if there are no rules that can be applied to the branch in a way that satisfies their constraints or that avoid adding duplicate formulae to the branch. If every branch of the tableau is saturated we say the tableau is saturated.

We will show later these new restrictions on these rules ensures that eventually any branch will be saturated, since the amount of times any of the term rules can be applied depends on the number of nominals and terms on the branch. We add the further restriction for universal quantification rules (just ($\neg\exists$) in our tableau method) where Q is a positive natural number:

In first-order tableau systems universal rules are the main culprit for non-termination. Since proofs are finite objects, any tableau proof for a formula involving universal quantification will close for some quantifier depth Q, the trouble being there is no general way to anticipate the value of Q as that would constitute a decision procedure for first-order logic. The main function of a universal rule in a tableau proof is to close the branch, so selecting a particular quantifier depth is a way of giving a branch every reasonable chance to close where it has a possible

$$\frac{@_s\neg(\exists x)\varphi}{@_s\neg\varphi[s{:}p/x]}\ (\neg\exists)^a$$

[a] p is any parameter on the branch (or any parameter in general if there are none on the branch yet), and the premise can produce only Q distinct conclusions.

application of a universal rule. While for model checking we actually will not need branches to close so long as they terminate, closed branches are helpful since we automatically do not have to check them for satisfiability.

4.2 Systematic tableau construction

Definition 4.2 ($\prec_\Theta$)**.** Where Θ is a tableau branch in the *FHTL* calculus, if a nominal a is introduced to the branch by application of of F or P to a premise $@_s\varphi$, we say a is *generated* by s on Θ and write $s \prec_\Theta a$.

Lemma 4.1. *Where Θ is a tableau branch in the FHTL calculus, the graph $G = (N_\Theta, \prec_\Theta)$ is the disjoint union of a finite set of well-founded and finitely branching trees.*

Proof. Each aspect is proved below:

- *Wellfoundedness of trees in G*

  We have that if $a \prec_\Theta b$ then the first occurence of a on Θ is before the first occurence of b, thus by induction any subset of N_Θ under the relation $\prec_\Theta$ has a least element and each tree in G is wellfounded.

- *G is a disjoint union of a set of trees*

  This follows from well-foundedness, the fact every nominal in N_Θ can be generated by at most one other nominal, and that every nominal in N_Θ must have one of the finitely many nominals in the root formula as an ancestor.

- *The set of trees is finite*

  If a nominal c occurs on a branch but not in the root formula then there is some nominal a which generated c, as a result c cannot be the root of a tree, and since there are only finitely many nominals in a root formula there can only be finitely many trees.

- *Each tree finitely branching*

  We show some tree $\mathcal{T}$ is finitely branching by showing that given a nominal a, there can only be finitely many distinct nominals b such that $a \prec_\Theta b$. Each nominal b such that $a \prec_\Theta b$ is generated by applying one of F, P to a premise of the form $@_aF\varphi$ or $@_aP\varphi$ respectively, where by our restrictions φ is not a nominal. Since there can only be finitely many root subformulae of the form of one of the the possible premises, where a is the prefix nominal in each case, only finitely many new nominals have been generated from a. Thus $\mathcal{T}$ is finitely branching.

□

Remark 4.1. In the restricted rules, a term can have no nominal among its prefixes more than once. Consequently where n is the number of nominals on the branch and $k(\leq n)$ is the length of a term's prefix string, the number of possible prefixed versions of a term are

$$\sum_{k=1}^{n} \frac{n!}{(n-k)!}$$

Lemma 4.2. *Using the restricted rules, minus any rules which generate new nominals on a branch (F and P), any FHTL tableau branch Θ saturates within a finite number of steps.*

Proof. The propositional rules, ($\neg F/P$), ($\exists$) and the @ rules, have only finitely many possible applications before they can no longer be applied on Θ. With the quantifier depth restriction ($\neg\exists$) can only be applied a finite number of times, and while **(sub)** can generate multiple conclusions since there are multiple partial substitutions for any formula, there are only finitely many partial substitutions possible. For **(ref)** and the term rules, we have at worst cases like **(:3)** and **(ref)** which depend on both the number of nominals and the number of terms on the branch for the number of possible applications before saturation. There are only finitely many nominals and terms on the branch at any given time, and by our remark, there are finitely many possible prefix strings for a given term. Thus there are only finitely many possible applications of **(ref)** to the branch. □

Definition 4.3 (*Tableau construction algorithm)***.** Let $@a_a\varphi$ be the formula whose local satisfiability (possibly validity) we are deciding. We inductively define a sequence of finite tableaux

$$\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$$

where each element of the sequence is embedded in all of its successors. We let $\mathcal{T}_0$ be the finite tableau consisting of the formula $@_a\neg\varphi$. Assuming the finite tableau $\mathcal{T}_n$ is defined, apply, where possible according to its restrictions, an arbitrary model checking tableau rule with the following restrictions.

- **Saturation**

  Do not apply a rule to a premise if all possible conclusions already occur on the branch.

- **(Loop check)**

  The rule (F) is not applied to a formula occurence $@_aF\varphi$ at a branch Θ if there is a nominal b such that $a \subseteq_\Theta b$, and simlarly for the rule (P).

Let $\mathcal{T}_{n+1}$ be the resulting tableau.

The saturation conditions prevent the algorithm from extending the tableau indefinitely with redundant formulae and forces the algorithm to use every applicable rule the maximum possible amount of times, i.e. this algorithm is "fair" (Fitting and Mendelsohn, 1998). The loop check condition is designed in the same spirit, preventing the algorithm from introducing nominals to the branch which would be redundant in their sharing identical information with some ancestor nominal already on the branch. Now we demonstrate the algorithm always terminates.

Theorem 4.3. *The systematic tableau construction algorithm terminates.*

Proof. Suppose by contradiction the algorithm does not terminate. Then the tableau is an infinite branch Θ. By lemma 4.1 the graph $(N_\Theta, \prec_\Theta)$ is a disjoint union of a finite set of finitely branching trees. Thus it has some inifinite branch

$$a_1 \prec_\Theta a_2 \prec_\Theta \ldots \prec_\Theta a_i \prec_\Theta \ldots$$

otherwise by lemmas 3.1 and 4.2 there would only be finitely many formulae on the branch, contradicting that it is infinite. For each i in the indices of this infinite chain of nominals, let Θ_i denote the initial segment of Θ up to but not including the first formula where the nominal a_{i+1} occurs. Thus one of (F) or (P) was applied to a formula in Θ_i which generated a_{i+1}. Let Γ_i be the set of all formulae which contain any root subformulae φ such that $@_{a_i}\varphi \in \Theta_i$ and let Δ_i be the set of all formulae which contain any root subformulae φ such that $@_{a_i}\neg\varphi \in \Theta_i$. Since there can only be finitely many sets of root subformulae, there are nominals in the infinite chain a_j, a_k where $j < k$ and $\Gamma_j = \Gamma_k$ and $\Delta_j = \Delta_k$. Since $j < k$ the first occurence of a_j on Θ_k must be before the first occurence of a_k on Θ_k. Additionally for any root subformula φ, if $@_{a_k}\varphi \in \Theta_k$ then $\varphi \in \Gamma_k$ and consequently $\varphi \in \Gamma_j$, but then $@_{a_j}\varphi \in \Theta_j$ which is an initial segment of Θ_k. Thus by definition 3.9 a_k is included in a_j with respect to Θ_k, and since the first occurence of a_j is before the first occurence of a_k we have $a_k \subseteq_\Theta a_j$. But this contradicts that (F) or (P) was applied on Θ_k resulting in the first formula containing an occurence of the nominal a_{k+1} since $a_k \subseteq_\Theta a_j$ would result in the loop check condition blocking the application. Thus the algorithm terminates. □

4.3 Node Annotation

The approach involves annotating each node of an open branch with the variable assignments in the model which satisfy the sentence at the node, building inductively from the terminal nodes. If the root formula of the tableau with at least one open branch can be annotated with non-empty set of variable assignments, then it is satisfiable in the model. If a tableau is closed then the root sentence $@_s\varphi$ is unsatisfiable. As a result if the root formula is of the form $@_s\neg\psi$ then this constitutes a proof of the validity of $@_s\psi$ by contradiction. We now view each node in the tableau graph as a pair $(@_s\varphi, \mathcal{V})$, of the formula at the node and the set $\mathcal{V}$ of variable assignments in our model $\mathfrak{M}$ which witness the formula. We define an annotation function $ann(@_s\varphi) = \mathcal{V}$ generally as:

$$ann(@_s\varphi) = \{g \mid \mathfrak{M}, I_{nom}(s) \vDash_g \varphi\}$$

If we have a branch that we know to be closed from the tableau construction (or discover it is closed during model checking as we will see) and $@_t\psi$ is the root of that branch, we define:

$$ann(@_t\psi) = \emptyset$$

For this reason during the annotation process we need not check formulae of the form $@_i j$ or $@_i(F/P)j$, or nodes not involving free variables since their truth or falsehood is independent of variable assignments. Terminal nodes will necessarily not contain quantifiers, double negation, nested prefix/satisfaction operators, or logical connectives, as a result they will consist only of equality or relation statements over extended terms possibly involving free variables. Now we can begin describing the dependency relationships between tableau rule conclusions and their premises.

- **Term rules:** In all cases we can see that for every rule with a premise the same variable assignments which satisfy the conclusion satisfies the premise. This is immediate for **(:2)**, **(:3)**, and **(:func)**, since they only involve closed terms, and are verified by checking definitions in all other cases.
- **@ rules:** For **(P/F–bridge)**, and **(P/F–trans)** all variable assignments trivially satisfy both the conclusions and premises. For **(nom)**, since by the first premise

$$I_{nom}(i) = I_{nom}(j)$$

  the variable assignments that satisfy the conclusion satisfy the second premise (the first premise being a nominal equivalence is trivally satisfied by all variable assignments). That the premise and conclusion of both ($@$) and ($\neg @$) are satisfied by the same variable assignments are verified by checking their satisfiability definitions.
- **Equality rules:** The only rule to check is **(sub)**, from which it follows immediately that the second premise is satisfied by the variable assignments which satisfy both the conclusion and the constraint given by the first premise, namely that $\overline{j{:}t} = \overline{k{:}s}$.
- **Quantifier rules:** For ($\exists$), we let C be set of variable assignments which satisfy the conclusion. Without loss of generality, where the premise is $@_s(\exists x)\varphi$, we see from the semantics of $\exists$ that P, the set of variable assignments which satisfies the premise can be defined as:

$$P = \{g \mid g \in C \text{ such that there is an } x\text{-variant } g' \text{ of } g \text{ at } s \text{ where } \mathfrak{M}, s \models_{g'} \varphi\}$$

  Similarly for ($\neg\exists$) let C be the set of variable assignments which satisfy the conclusion. Without loss of generality, where the premise is $@_s\neg(\exists x)\varphi$, we see from the semantics of $\neg\exists$ that P, the set of variable assignments which satisfies the premise can be defined as:

$$P = \{g \mid g \in C \text{ such that for every } x\text{-variant } g' \text{ of } g \text{ at } s \text{ we have } \mathfrak{M}, s \models_{g'} \neg\varphi\}$$

- **Modal rules:** For F and P, that the the premise and the second conclusion are satisfied by the same variable assignments follows from checking the definitions. Similarly for the first premise and the conclusion of $\neg P$ and $\neg F$.
- **Propositional rules:** For ($\vee$) given the semantics of the premise, the union of the sets of variable assignments satisfying each of the conclusions satisfies the premise. Similarly for ($\neg\vee$), except we take the intersection of the variable assignment sets. That the premise and conclusion of ($\neg\neg$) are satisfied by the same set of variable assignments follows from checking the semantics.

Remark 4.2. We note that any annotation of variable assignments for a node will be finite since any formula will have only finitely many variables, and the model's domain itself is finite.

Lemma 4.4. *Node annotation for a model checking tableau over a finite model is decidable.*

Proof. We give an inductive argument, where the base cases are closed terms of the extended language. First we consider closed terms. Since any model checking tableau is finite, it has only finitely many function symbols, constants, and parameters. Since parameters are not free variables but stand in for them, we will handle them as follows: Recalling every parameter is of the form $j{:}\ldots{:}i{:}p$ with a string of at least one distinct prefixes (i is called the *immediate prefix* in the example):

1. Proceeding with annotation as normal, ignore any parameter until reaching the lowest equality or relation statement where it appears in the conclusion, then solve for the values in $D_{I_{nom}(i)}$ for which the parameters would satisfy the formula. Clearly this process terminates since for any nominal i, $D_{I_{nom}(i)}$ is finite.
2. From here on, we treat the sets of parameters which satisfy a formula as we would the set of variable assignments which would satisfy a formula with the exception of quantifier rules which we will cover later. In particular, for a case of ($\vee$) as in $@_s(\varphi[i{:}p] \vee \psi[i{:}p])$ we take the union of the parameter sets for $@_s\varphi[i{:}p]$ and $@_s\psi[i{:}p]$, similary taking the intersection in the case of ($\neg\vee$)

The proof of termination for computing these parameter annotations for each node is essentially the same as the proof of termination for variable assignment annotations, so without loss of generality we omit the former and proceed with the latter. The model has only finitely many worlds/times and a finite domain for each world/time, consequently computing the value of a closed term is guaranteed to terminate and the annotation consists of every possible variable assignment (any implementation of this process would represent the totallity of variable assignments via a flag which would consume only constant space, similarly for parameter annotations until we solve for them). Similarly, computating annotations for relation and equality statements over closed terms terminates. Since the @ rules, modal rules, and ($\neg\neg$) do not involve any operations on the set of variable assignments satisfying the conclusions, by induction these node annotations terminate. Assume by induction computing the annotations for the conclusions of ($\vee$) and ($\neg\vee$) terminates. Recalling 4.2, we see that ($\vee$) and ($\neg\vee$) involve only union and intersection of finite sets respectively. For **(sub)** by we assume by induction that computing the annotations for the first premise and the conclusion terminates, since they are finite, computing their intersection terminates and we have the annotation for the second premise. Finally assume by induction that computing the annotations for the conclusions of ($\exists$) and ($\neg\exists$) terminates. For ($\exists$), assuming without loss of generality the premise is of the form $@_s(\exists x)\varphi$, recalling 4.2 again, we check each variable assignment from a finite set whether it has an x-variant at s that satisfies the premise stripped of its quantifier. But since we have parameter annotations, it suffices to check if the conclusion's parameter annotation is non-empty, since that ensures for any variable assignment an appropriate x-variant could be constructed. As a result if the parameter annotation is non-empty then the variable assignment annotation for the premise is the same as for the conclusion, clearly this step of the process terminates. Similarly for ($\neg\exists$), in determining whether the variable assignments in the annotation for the conclusion satisfy the premise, it suffices the annotation for the relevant parameter, say $i{:}p$ is the same size as $D_{I_{nom}(i)}$. If they are the same size then the variable assignments annotation for the premise is the same as for the conclusion, and since $D_{I_{nom}(i)}$ is finite clearly this terminates. □

Theorem 4.5. *Finite model checking for FHTL via restricted tableau is decidable.*

Proof. This follows immediately from theorem 4.3 and lemma 4.4. □

5 AMR Interpretation in *FHTL*

In this section we present an approach for translating AMR annotated for quantification and scope along with tense and aspect into *FHTL*. In this work we do not make use of aspect but we discuss a possibility of how to handle it later. Where AMR interpretations given in Lai et al. (2020) and Bos (2016) translate AMR into first-order predicate logic via a direct and stateless transformation on the AMR, we elected to demonstrate a stateful approach using pseudocode, since it is clearer how to extract time and tense information this way, and how to implement this approach, which we hope to do soon. The core of our interpretation process is taken from Pustejovsky et al. (2019), since the `scope` node dictates order and and application of the interpretation of the subgraphs in the AMR. We assume the AMR's temporal information is attached to the concept at the `scope`'s `pred` node, and integrate that into the translation to achieve an interpretation in *FHTL* instead of first-order predicate logic.

5.1 Examples

(1) a. Every computer will be located at a desk.
b.
```
(s / scope
      :pred (b / be-located-at-91 :ongoing - :complete + :time (a / after :op1 (n
/ now))
        :ARG0 (c / computer)
        :ARG1 (d / desk
           :quant (e / every)))
      :ARG0 d
      :ARG1 c)
```
c. $@_{now}(\forall y)[\text{desk}(y) \to (\exists x)[\text{computer}(x) \wedge F(\text{be-located-at-91}(x,y))]]$
d. Also acceptable but with a subtle difference *It will be the case that every computer will be located at a desk*: $@_{now}F((\forall y)[\text{desk}(y) \to (\exists x)[\text{computer}(x) \wedge \text{be-located-at-91}(x,y)]])$

(2) a. Carl filled out the forms and everyone will submit them tomorrow.
b.
```
(a / and
   :op1 (s / scope
      :pred (f / fill-out-03 :ongoing - :complete + :time (b / before :op1 (n / now))
          :ARG0 (p / person
              :name (n2 / name
                  :op "Carl"))
          :ARG1 (f2 / form))
      :ARG0 p
      :ARG1 f2)
   :op2 (s2 / scope
      :pred (m / submit-01 :ongoing - :complete + :time (a2 / after :op1 n)
          :ARG0 (p2 / person
              :mod (a3 / all))
          :ARG1 f2)
      :ARG0 f2
      :ARG1 p2))
```

c. Technically correct:
$@_{now}(\exists x)[\texttt{form}(x) \wedge P(\texttt{fill-out-03}(\texttt{Carl}, x))) \wedge @_{now}(\exists x)[\texttt{form}(x) \wedge (\forall y)[\texttt{person}(y) \rightarrow F(\texttt{submit-01}(y, x))]]$
d. Correct with regard to reentrance:
$@_{now}(\exists x)[\texttt{form}(x) \wedge P(\texttt{fill-out-03}(\texttt{Carl}, x)) \wedge (\forall y)[\texttt{person}(y) \rightarrow F(\texttt{submit-01}(y, x))]]$

5.2 Extraction Steps

With the chosen annotation, the root node can consist of either a logical connective (`and`, `or`, or `cond`) linking two or more AMR graphs, or a `scope` node with its following predicate and arguments. In the former case we interpret the arguments and join them together via the meaning of the connective, e.g. for:

(3)
```
(o / or
    :op1 (...)
    :op2 (...)
    :op3 (...))
```

The meaning in *FHTL* is $[\![\texttt{:op1}]\!] \vee [\![\texttt{:op2}]\!] \vee [\![\texttt{:op3}]\!]$ In the case where the root node of the AMR is `scope`, the order of interpretation is specified. For instance in 1, the order of interpretation is $[\![\texttt{ARG0}]\!]([\![\texttt{ARG1}]\!]([\![\texttt{pred}]\!]))$. Finally, we have that the meaning of every AMR with a `scope` root will be prefixed by a satisfaction operator with a nominal indicating the time of reference provided in the `:time` relation (if that time is "`now`" then we let the nominal indicate the document creation time where available).

5.3 General Extraction Algorithm

Algorithm 1: Basic transformation into *FHTL* clauses and connectives.

```
Input: AMR sentence
Output: FHTL formula
Def InterpretEntry(AMR):
    root = Root(AMR)
    now = current date/time
    if root ∈ {and, or, cond} then
        connective = filter(root, {∧, ∨, →})
        clauses = []
        for op ∈ Children(root) do
            append(clauses, InterpretClause(op))
        end
        return $@_{now}$ join(connective, clauses)
    end
    return $@_{now}$ InterpretClause(root)

Def InterpretClause(AMR):
    time = Time(AMR)
    nominal = Reference(time)
    tense = Tense(time)
    pred = Pred(AMR)(tense)
    $Arg_0$, $Arg_1$ = GetArgs(AMR)
    return $@_{nominal}$ Apply($Arg_0$, Apply($Arg_0$, pred))
```

`InterpretEntry` and `InterpretClause` perform essentially what we discussed in 5.2. We should note that in `InterpretClause` that the interpreted predicate takes the tense operator as an argument. We do not provide definitions for `Tense` and `Reference` since they are issues of checking nodes in the AMR graph. Similarly, `Pred` and `GetArgs`, are issues of straightforward graph traversal and modifications to the standard interpretation processes which we will show later and we do not provide them.

Algorithm 2: Supporting definitions.

```
Def Apply(pred_1,pred_2):
    return $\lambda\varphi$.pred$_1$($\lambda\psi$.pred$_2$($\lambda\gamma.\varphi(\psi(\gamma))$))

Def InterpretPred(UnaryPred):
    /* Propositional modifiers are handled differently than predicates. */
    if isPropMod(name(UnaryPred)) then
        return InterpPropMod(UnaryPred)
    end
    if hasMods(UnaryPred) then
        mods = [name(UnaryPred)]
        for mod $\in$ Children(UnaryPred) do
            append(mods, name(mod)($x$))
        end
        FinalPred = $\lambda x$. join(mods, $\wedge$)
    end
    else
        FinalPred = $\lambda x$.name(UnaryPred)($x$)
    end
    if Quant(UnaryPred) == :all then
        return $\lambda k.\forall x$[FinalPred($x$) $\rightarrow k(x)$]
    end
    else
        return $\lambda k.(\exists x)$[FinalPred($x$) $\wedge k(x)$]
    end

Def InterpPropMod(PropMod):
    if PropMod == not then
        return $\lambda p.\lambda k.\neg p(k)$
    end
```

Apply is standard continuation based application of predicates (Van Eijck and Unger, 2010). InterpretPred handles AMR concepts possibly with nested relations to other concepts, letting us extract $@_t(\forall x)((dog(x) \wedge friendly(x)) \rightarrow P(bark(x)))$ from *Every friendly dog barked* rather than recursively obtaining a Neo-Davidsonian approach as in Lai et al. (2020). InterpPropMod handles AMR concepts which modify entire sentences/propositions such as negation and modality.

6 Future Work

6.1 $\downarrow$ and Quantification over Nominals

Hansen (2007) includes treatment of two operators which we have omitted, namely $\downarrow$ and E. $\downarrow$ binds a nominal to the point of evaluation, i.e. $@_a \downarrow w.\varphi$, $\downarrow w.\varphi$ is the case at a, if and only if φ is the case at w ($@_w\varphi$) when a refers to w. This feature allows to represent statements such as *Caroline is the current chief architect* via a formula of the form $\downarrow x.(c = @_x a)$ where c is a rigid designator (for us a constant) denoting *Caroline* and a is a non-rigid designator (for us a unary function symbol) denoting *chief architect* (Blackburn and Marx, 2002). Intuitively E is existential quantification over nominals (and $\neg E \neg \equiv A$ is universal quantification over nominals), i.e. $@_s E\varphi$ if there is some nominal j such that $@_j\varphi$. Quantification over nominals gives us an entry into handling aspect in addition to tense. For instance, the :stable + and :stable − aspects from Donatelli et al. (2018) can be modeled via A and $\downarrow$ respectively. The different modes of :ongoing ostensibly could be modeled with operators from computation tree logic such as finally and until (May and Schmitt, 1996). :habitual is the most problematic case, since this would require generalized quantification over nominals, making inference and model checking very expensive if nothing else.

6.2 Implementing Theorem Proving and Model Checking

Now that we have a general approach for automated inference and model checking of first-order hybrid logic, there is the question of implementation. HTab (Hoffmann and Areces, 2009) provides an implementation of $\mathcal{H}(@, \mathsf{A})$, (propositional hybrid logic with quantification over nominals) which does not natively provide a way to reason with P or first-order quantification. While it may be possible to extend HTab to accomodate these, it might be more productive to consider theorem provers for first-order modal logic utilizing tableau such as leanTAP (Beckert and Posegga, 1995) and extending them to accomodate hybrid logic's satisfaction operator and the other aspects of *FHTL*. MDK-verifier [2] is the only modal model checker we are aware of, and currently only works with

[2]http://www.cril.univ-artois.fr/~montmirail/mdk-verifier/

propositional K.

6.3 The Future of AMR and Parsing for Semantic Features

AMR parsing is an area which receives a great deal of attention, we are not aware however of many efforts in AMR parsing towards accomodating extensions with these semantic features which enable any kind of descriptive direct inference. Indeed, whether any research in automated reasoning over AMR and its extensions is of practical relevance depends entirely on whether AMR parsers can accomplish this in a resource effective way. Determining where and to what extent AMR parsing for extensions is effective is a worthwhile pursuit for AMR researchers in general, whether or not they are concerned with the semantic descriptiveness of these extensions or their suitability for inference.

7 Conclusion

We have demonstrated how the core aspects of AMR along with extensions for scope, quantification, and tense, can be interpreted in first-order hybrid tense logic, and that *FHTL* affords reasoning for these translated AMR sentences, through its general tableau method which acts as a proof procedure for *FHTL* sentences and its terminating tableau method which acts as a model checker. We have discussed the possibility of handling aspect and what is required for it, and some possible strategies for developing an implementation of *FHTL* or another first-order hybrid logic variant. There are further questions in these areas of exploration of both theoretical and pragmatic approaches to optimization of these techniques, as well as investigating how AMR parsing can become both robust and sensitive enough to realize the goals of this research.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544.

Bernhard Beckert and Joachim Posegga. 1995. leantap: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358.

Patrick Blackburn and Klaus Frovin Jørgensen. 2012. Indexical hybrid tense logic. *Advances in Modal Logic*, 9:144–60.

Patrick Blackburn and Maarten Marx. 2002. Tableaux for quantified hybrid logic. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 38–52, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jürgen Bohn, Werner Damm, Orna Grumberg, Hardi Hungar, and Karen Laster. 1998. First-order-ctl model checking. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 283–294. Springer.

Thomas Bolander and Torben Braüner. 2006. Tableau-based Decision Procedures for Hybrid Logic. *Journal of Logic and Computation*, 16(6):737–763.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Johan Bos. 2016. Squib: Expressive power of Abstract Meaning Representations. *Computational Linguistics*, 42(3):527–535.

Lucia Donatelli, Michael Regan, William Croft, and Nathan Schneider. 2018. Annotation of tense and aspect semantics for sentential AMR. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 96–108, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Melvin Fitting. 1988. First-order modal tableaux. *Journal of Automated Reasoning*, 4(2):191–213.

Melvin Fitting and Richard L Mendelsohn. 1998. *First-Order Modal Logic*, volume 277. Springer Science & Business Media.

Rod Girle. 2014. *Modal logics and philosophy*. Routledge.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *AKBC '13*.

Jens Ulrik Hansen. 2007. A tableau system for a first-order hybrid logic. In *Proceedings of the International Workshop on Hybrid Logic (HyLo 2007)*, pages 32–40.

Guillaume Hoffmann and Carlos Areces. 2009. Htab: a terminating tableaux system for hybrid logic. *Electronic Notes in Theoretical Computer Science*, 231:3–19. Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).

Kenneth Lai, Lucia Donatelli, and James Pustejovsky. 2020. A continuation semantics for Abstract Meaning Representation. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 1–12, Barcelona Spain (online). Association for Computational Linguistics.

Wolfgang May and Peter H Schmitt. 1996. A tableau calculus for first-order branching time logic. In *International Conference on Formal and Applied Practical Reasoning*, pages 399–413. Springer.

Markus Müller-Olm, David Schmidt, and Bernhard Steffen. 1999. Model-checking. In *International Static Analysis Symposium*, pages 330–354. Springer.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Arthur N Prior and Per FV Hasle. 2003. *Papers on time and tense*. Oxford University Press on Demand.

James Pustejovsky, Ken Lai, and Nianwen Xue. 2019. Modeling quantification and scope in Abstract Meaning Representations. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 28–33, Florence, Italy. Association for Computational Linguistics.

Lenhart K. Schubert. 2015. What kinds of knowledge are needed for genuine understanding? In *IJCAI 2015 Workshop on Cognitive Knowledge Acquisition and Applications (Cognitum 2015)*.

Jan Van Eijck and Christina Unger. 2010. *Computational semantics with functional programming*. Cambridge University Press.