Formalization of AMR Inference via Hybrid Logic Tableaux

Eli Goldner

July 22, 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 various 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 *FHTL* (first-order hybrid tense logic), and demonstrate *FHTL*'s soundness, completeness, and decidability.

1 Introduction

2 Related Work

2.1 Semantic Features in AMR and Possbility of Inference

Separating argument structure in AMR from logical structure, enables translation from AMR to DRT (Bos, 2020)

AMR expressivity without recurrent variables (and with no more than one universal quantifier per sentence) are in the decidable two-variable fragment of first-order logic (Bos, 2016)

Extension of sentential AMR to incorporate a coarse grained treatment of tense and aspect (Donatelli et al., 2018)

Continuation based semantics for translating AMR into first-order logic in a way that preserves projection phenomena such as quantification, negation, bound variables, and donkey anaphora, which better affords inference than other first-order logic semantics for AMR (Lai et al., 2020). This uses a neo-davidsonian representation for the target first-order logic semantics.

3 Merging Quantified Hybrid Logic and Indexical Hybrid Tense Logic

3.1 Background

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.

3.2 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). Hansen (2007)

from
Blackburn and Marx (2002)

3.3 Basic Hybrid Tense Logic

Blackburn and Jørgensen (2012)

4 First-order Hybrid Tense Logic - Syntax and Semantics

The syntax of *FHTL* is identical to *FHL* as given in Hansen (2007) except uses of $\downarrow$ as in $\downarrow w.\varphi$ are omitted along with $\Box$ and $\Diamond$ as in $\Box\varphi$ and $\Diamond\varphi$. $\Box$ and $\Diamond$ are replaced by their semantic equivalents F and G and their temporal duals P and H are added.

Atomic formulae are the same as in *FHL*, symbols in NOM and SVAR together with first-order atomic formulae generated from the predicate symbols and equality over the terms. Thus complex formulae are generated from the atomic formulae according to the following rules:

$$\neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \exists x\varphi \mid \forall x\varphi \mid F\varphi \mid G\varphi \mid P\varphi \mid H\varphi \mid @_n\varphi$$

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:

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

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 on the level of 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.
- $I_t(P) \in D^k$ where P is a k-ary predicate symbol.

Notice we've relaxed the requirement that $I_t(c) = I_{t'}(c)$ for c a constant and $t, t' \in T$, since the interpretation of the constant need not exist at both times. This permits us to distinguish between the domain of a frame and the domain of a time/world, in a way that prevents a variable x from failing to refer at a given time/world, 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. previous and future 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 an assignment g, the interpretation of terms t denoted by $\bar{t}$ is defined as:

- $\overline{x} = g(x)$ for x a variable.
- $\overline{c} = I_t(c)$ for c a constant and some $t \in T$.
- For q a unary function symbol:
  - For n a nominal:
    $$\overline{@_n q} = I_{I_{nom}(n)}(q)$$
  - For n a state variable:
    $$\overline{@_n q} = I_{g(n)}(q)$$

Finally we say an assignment g' is an x-variant of g if g' and g on all variables except possiblly x. In particular, we say g' is an x-variant of g at t, a time, if g' and g on all variables except possiblly x and $g'(x) \in D_t$. We omit definitions for $\wedge$, $\rightarrow$, H, G, and $\forall$, since they can be defined in terms of the other rules. Given a model $\mathfrak{M}$, a variable assignment g, and a state s, the inductive definition of $\mathfrak{M}, s \vDash_g \varphi$ is:

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

4.1 The Tableau Calculus

Following Fitting and Mendelsohn (1998) we assume for each nominal or state variable s, there is an infinite list of parameters, 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. Informally we write p_s to indicate a parameter is associated with a nominal/state variable s.

4.2 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 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) en route to completeness 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 terminating tableau system is for finite model checking.

Definition 4.1 (**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*.

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. The question of when a tableau indicates satisfiability of the root formula leads us to our next definition.

Definition 4.2 (**Saturation).** A tableau branch is *saturated* if no more rules can be applied to the branch in a way that satisfies their constraints. If every branch of the tableau is saturated we say the tableau is saturated.

Definition 4.3 (**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 renaming of free variables and substitution of free variables in φ for grounded terms.
2. φ is of the form $\neg\chi$ where χ is a subformula of ψ modulo renaming of free variables in χ for grounded terms.

Altering the definition to allow grounded terms being substituted for free variables ensures compatibility of the following proofs with the quantifier and term 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.

Propositional rules:

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

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:

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

Equality rules:

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

@ rules:

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

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

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

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

***FHTL* term rules:**

$$\frac{@_ik_1{:}t = k_2{:}s}{@_ik_1{:}t = k_2{:}s}\ \textbf{(:1)} \qquad \frac{@_ij}{@_ki{:}t = j{:}t}\ \textbf{(:2)}^a \qquad \frac{}{@_ik{:}j{:}t = j{:}t}\ \textbf{(:3)}^a$$

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

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

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

Lemma 4.1 (Subformula Property). *Where Θ is a tableau branch in the FHTL calculus, and a formula $@_s\varphi$*

occurs on Θ 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 root subformula.

Proof. This is verified by checking the tableau rules. □

Theorem 4.2. *Let $@_a Fb$ 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 $@_a Pb$.*

Proof. This is verified by checking each of the tableau rules, in some cases it is necessary to invoke lemma 4.1. □

Definition 4.4 ($\prec_\Theta$)**.** 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 $s \prec_\Theta a$. We write $\prec^*_\Theta$ to denote the reflexive and transitive closure of $\prec_\Theta$.

Definition 4.5 (N_Θ)**.** The set of nominals and state variables which occur on Θ is written N_Θ

Lemma 4.3. *Where Θ is a tableau branch in the FHTL calculus, the graph $G = (N_\Theta, \prec_\Theta)$ is a wellfounded finitely branching tree.*

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 tree*

  Every nominal in N_Θ can be generated by at most one other nominal, and every nominal in N_Θ must have one of the finitely many nominals in the root formula as an ancestor.

- *G is finitely branching*

  We show G 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 $@_i F\varphi$ or $@_i P\varphi$ respectively, where by our restrictions, either φ is not a nominal, or the entire premise is a root subformula. Since there can only be finitely many root subformulae of the form of one of the the possible premises, where i is the prefix nominal in each case, only finitely many new nominals have been generated from i. Thus G is finitely branching.

□

Lemma 4.4. *Where Θ is a tableau branch in the FHTL calculus, Θ is infinite if and only if there exists an infinite chain of nominals and state variables $a_1 \prec_\Theta a_2 \prec_\Theta \ldots \prec_\Theta a_n \prec_\Theta \ldots$*

Proof. Since the structure of the formulae and tableau rules are not involved in the proof from Bolander and Braüner (2006) holds here as well.

□

Definition 4.6 ($\subseteq_\Theta$)**.** 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 4.7 ($\sim_\Theta$)**.** 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 $\sim_\Theta$.

Definition 4.8 ($W, \approx$)**.** 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 4.9. 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. 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 4.5. *If t is a term of the extended language that contains no variables and i is a nominal then we have $[\![t]\!]^{\mathfrak{M},g}_{[i]} = \overline{i{:}t}$ 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

$$[\![t]\!]^{\mathfrak{M},g}_{[i]} = [\![t]\!]^{\mathfrak{M}}_{[i]} I_{[i]}(t) = \overline{i{:}t}$$

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:

$$[\![u{:}s]\!]^{\mathfrak{M},g}_{[i]} = [\![s]\!]^{\mathfrak{M},g}_{[u]} \overset{*}{=} \overline{u{:}s} \overset{\textbf{(:3)}}{=} \overline{i{:}u{:}s}$$

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

Lemma 4.6. *Assuming the branch Θ is open, for any formula $@_a\varphi$ which contains only nominals from W, the following two statements hold:*

- $@_a\varphi \in \Theta \Longrightarrow \mathfrak{M}, [a] \vDash_g \varphi$ *for some (all) variable assignment g*
- $@_a\neg\varphi \in \Theta \Longrightarrow \mathfrak{M}, [a] \nvDash_g \varphi$ *for some (all) variable assignment 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 $\mathfrak{M}, [c] \vDash_g \psi$. 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 $R[a][b]$. By theorem 4.2 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
- *ψ not a nominal*

□

For our task of AMR inference, we are not 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 model 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 predicate symbols) can be referenced. In particular, we have a case 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 $\mathfrak{M}, t \vDash_g \varphi$ (Müller-Olm et al., 1999).

Here we develop a restriction of our method for tableau construction, which ensures termination at the cost of completeness. This compromise only has positive ramifications for model checking since it does not particularly benefit from completeness of a tableau method but termination is critical to decidability. 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 **?**.

4.3 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 consequent.
[f] f is an n–ary function symbol and $t_1, ..., t_n$ are closed terms with i not among their prefixes.

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:

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

[a] p is any parameter and the premise can produce only Q conclusions.

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 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.

4.4 Systematic tableau construction

Definition 4.10 (*Tableau construction algorithm)***.** Where $@_a a_a \varphi$ is the formula whose validity we are deciding. We inductively define a sequence of finite tableaux $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ each 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 $\neg @ a_a \varphi$. Assuming the finite tableau $\mathcal{T}_n$ is defined. If possible, apply an arbitrary *FHTL* tableau rule with the following restriction.

- **(Loop check)** The rule F is not applied to a formula occurence $@_a F\varphi$ 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.

Theorem 4.7. *The systematic tableau construction algorithm terminates.*

Proof. Suppose by contradiction ... □

4.5 Node Annotation

The approach involves annotating each node of an open branch with the variable assignments in the model which witness the formula 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 formula $@_s \varphi$ 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 $ann(@_s \varphi) = \mathcal{V}$ beginning with terminal nodes:

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

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 witness the conclusion witness the second premise (the first premise being a nominal equivalence is trivally witnessed 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 premise and conclusion are witnessed by the same variable assignments.
- **Quantifier rules:** These are the first and only cases where it is perhaps not obvious what to do. For ($\exists$), we consider the variable assignment functions which witness the conclusion along with some domain element for which the new parameter in the conclusion witnesses the conclusion. Taking this same set of assignments with the exception of fixing the value of the quantified variable in the premise to the domain element assigned to the parameter, we can see that this set of variable assignments satisfies the premise. For ($\neg\exists$), we cannot rely on any number of conclusions to give us a set of variable assignments which will satisfy the premise, since distinct parameters need not have distinct values. As a result we simply check whether each assignment of the quantified variable satisfies the premise. If every assignment satisfies the premise then we can proceed, if not, then we have effictively have a parameter for which we can contradict the conclusion of the rule thereby closing the branch.
- **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.

Lemma 4.8. *Node annotation for a model checking tableau over a finite model is decidable.*

Theorem 4.9. *Finite model checking for FHTL is decidable.*

5 AMR Interpretation in *FHTL*

5.1 Examples

Easier 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. $@_{now}\forall y[\text{desk}(y) \rightarrow \exists[\text{computer}(x) \wedge F\text{be-located-at-91}(x, y)]]$
d. Also acceptable $@_{now}F\forall y[\text{desk}(y) \rightarrow \exists[\text{computer}(x) \wedge \text{be-located-at-91}(x, y)]]$

Harder examples:

(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 (maybe requires the passive for singular case?):
$@_{now}\exists x[\texttt{form}(x) \wedge P\texttt{fill-out-03}(\texttt{Carl}, x) \wedge \forall y[\texttt{person}(y) \rightarrow F\texttt{submit-01}(y, x)]]$

The difficulty of the following is due to requiring inference of absent arguments.

(3) a. It was impossible not to notice the car.
b.

```
(s / scope
      :pred (p / possible-01
         :ARG0 (n / notice-01 :ongoing - :complete + :time (b / before :op1 (n2 /
now))
            :polarity (n3 / not)
            :ARG1 (c / car)
         :polarity (n4 / not))
      :ARG0 n4
      :ARG1 p))
```

c. Incorrect, since `notice-01` is a two-place predicate:
$@_{now}\neg\texttt{possible-01}(\exists x[\texttt{car}(x) \wedge \neg P\texttt{notice-01}(x)])$
d. Correct:
$@_{now}\neg\texttt{possible-01}(\exists x[\texttt{car}(x) \wedge \neg\forall y[\texttt{person}(y) \rightarrow P\texttt{notice-01}(x, y)]])$
e. Alternative, the particular car as a constant:
$@_{now}\neg\texttt{possible-01}(\neg\forall x[\texttt{person}(y) \rightarrow P\texttt{notice-01}(\texttt{car}, y)]])$

5.2 Extraction Steps

With the chosen annotation, the root node can consist of either a logical connective (`and`, `or`, or `cond`) linking two AMR graphs, or a `scope` node with its following predicate and arguments.

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* $\in \{\texttt{and}, \texttt{or}, \texttt{cond}\}$ **then**
    connective = filter(root, $\{\wedge, \vee, \rightarrow\}$)
    clauses = []
    for *op* $\in$ *Children(root)* **do**
      append(clauses, `InterpretClause(op)`)
    end
    return $@_{\text{now}}$ join(connective, clauses)
  end
  return $@_{\text{now}}$ `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 $@_{\text{nominal}}$ `Apply` (Arg_0, `Apply` (Arg_0, pred))

Algorithm 2: Supporting definitions.

Input: AMR sentence
Output: *FHTL* formula
Def `Apply(`$pred_1$,$pred_2$`)`**:**
  return $\lambda\varphi.\text{pred}_1(\lambda\psi.\text{pred}_2(\lambda\gamma.\varphi(\psi(\gamma))))$

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 $\lambda k.\forall x[\text{FinalPred}(x) \rightarrow k(x)]$
  end
  else
    return $\lambda k.\exists x[\text{FinalPred}(x) \wedge k(x)]$
  end

Def *InterpPropModPropMod*:
  if *PropMod* == `not` **then**
    return $\lambda p.\lambda k.\neg p(k)$
  end
  `/* Given the new sematics of` $F/\Diamond$ `*/`
  `/* we adopt the naïve way of treating possible-01 */`
  `/* as a predicate symbol. */`
  if *PropMod* == `possible-01` **then**
    return $\lambda p.\lambda k.$possible-01$(p(k))$
  end

6 Future Work

6.1 $\downarrow$ and Quantification over Nominals

Main points, at the cost of undecidability with adding $\downarrow$ some additional things can be done, and at the cost of the integration of generalized quantifiers you can ostensibly handle even things like habitual aspect.

6.2 AMR Reentrancy and Non-Temoral Nominals

There are some difficulties with maintaining the usual notion of possible worlds being maximal with this idea, but there seems to be a direct sympathy between the predicative core of an AMR sentence and in general reentrancy of the nodes with the idea of a nominal as a "point of view" rather than the "name" of a world. Maybe things like epistemic logic could be helpful here.

6.3 Automated Inference and HTab

HTab (Hoffmann and Areces, 2009) provides an implementation of $\mathcal{H}(@, \mathsf{A})$, which does not natively provide a way to reason with P, H, or first-order quantification. The effort required in making the needed changes to handle these remains to be determined.

6.4 The Future of AMR and Parsing for Semantic Features

To what extent can current AMR parsers extract the needed semantic features to make full use of automated inference? Of UMR, Dialogue-AMR, and the AMR annotation variants we've used, which logistically has the best outlook?

7 Conclusion

We have demonstrated how the core aspects of AMR along with annotations for scope, quantification, and tense, can be accomodated in first-order hybrid tense logic, and that *FHTL* better affords reasoning and inference than AMR, given its general tableau method which acts as a proof procedure for *FHTL* sentences and its terminating tableau methond which acts as a model checker.

References

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.

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

Johan Bos. 2020. Separating argument structure from logical structure in AMR. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 13–20, Barcelona Spain (online). Association for Computational Linguistics.

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

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.

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

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.