Formalization of AMR Inference via Hybrid Logic Tableaux

Eli Goldner

July 6, 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 Merging Quantified Hybrid Logic and Indexical Hybrid Tense Logic

1.1 Background

1.2 First-order Hybrid Logic

Hansen (2007)
from
Blackburn and Marx (2002)

1.3 Basic Hybrid Tense Logic

Blackburn and Jørgensen (2012)

2 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:

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

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:

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

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 $\overline{t}$ 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:

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

2.1 The Tableau Calculus

For Nom we have the constraint that if the premise $@_t\varphi$ are of the form $@_tXc$ where $X \in \{F, P, \neg G, \neg H\}$ and c is a nominal or state variable, then $@_t\varphi$ is a root subformula. Similarly for Nom^{-1} and the premise $@_s\varphi$.

In all rules in **??**, the nominal a is new to the branch. We have the additional constraint that if φ in the premise is a nominal or state variable, then the premise must be a root subformula in order for the rule to be applicable.

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.

We also introduce the notion of a grounded term. A grounded term is either a first-order constant, a parameter, or a grounded definite description, i.e. a term of the form $@_nq$ for n a nominal and q a unary function symbol.

In $(\exists)$, $s{:}p$ is a parameter associated with the nominal s, with the requirement that it is new to the branch. Since parameters are never quantified over, $s{:}p$ is free in $\varphi[s{:}p/x]$ but refers the same way as a constant. In $(\neg\exists)$, t is any parameter or term without variables which exists at s. Following Fitting (1988) to ensure termination of tableau construction we borrow the notion of quantifier depth

2.2 Soundness and completeness

The proof of soundness and completeness of *FHL* in Hansen (2007) uses the notions of $\Diamond$*–completeness* and $\exists$*–completeness*. As a result we substitute $\Diamond$–completeness with the equivalent notion F*–completeness* and add the notion of P*–completeness* where the $\exists$–completeness stays the same as before. Thus set of @-formulae (formulae in the extended language (including parameters) of the form $@_i\varphi$) is F–complete if:

$$@_iF\varphi \in S \Longrightarrow @_iFj \text{ or } @_jPi, @_j\varphi \in S, \text{ for some nominal } j$$

A set of @-formulae is P–complete if:

$$@_iP\varphi \in S \Longrightarrow @_iPj \text{ or } @_jFi, @_j\varphi \in S, \text{ for some nominal } j$$

Soundness and completeness for for *FHTL* follows straightforward modification of Hansen's proof of soundness and completeness for *FHL* in a way that reflects the new formulae of the form $@_iP\varphi$.

2.3 Tableau Construction

Definition 2.1 (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 2.2 (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.

For the unconstrained tableau rules, completeness gives us that an open saturated tableau is satisfiable. Since the constrained rules only guarantee termination in the case of finite models, we are not compromising completeness for termination we have the same guarantee for open saturated tableaux. As a result, given termination of the tableax and finite models, we can use the tableaux to check whether the root formula is satisfied in the model, in particular the process is decidable.

The proof of termination for every tableau construction is adapted from the proof given in Bolander and Blackburn (2009) for the termination of the tableau construction algorithm for $\mathcal{H}(@)$.

Definition 2.3. When a formula $@_s\varphi$ occurs in a tableau branch Θ we will write $@_s\varphi \in \Theta$, and say φ is true at s on Θ or s makes φ true on Θ.

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:

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

Quantifier rules:

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

Equality rules:

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

@ rules:

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

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

***FHTL* term rules:**

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

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

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

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

Definition 2.4. Given a tableau branch Θ and a nominal s the *set of true formulae* at s on Θ, is written $T^\Theta(s)$ and defined as follows:

$$T^\Theta(s) = \{\varphi \mid @_s\varphi \in \Theta\}$$

Definition 2.5 (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 grounded terms.

2. φ is of the form $\neg\chi$ where χ modulo substitution 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.

Definition 2.6 (**Accessibility formula).** A formula of the form $@_sFt$ or $@_sPt$ on Θ is called an *accessibility formula* if it is the first conclusion of an application of F or P.

Definition 2.7 (**Term equality formula).** A formula of the form $@_it = s$ where s and t are extended terms is called a term equality formula if it is a an immediate conclusion of **(ref)**, **(:3)**, or **(:func)**, or is generated from such a formula by one or more applications of **(:1)**, **(:2)**, **(:fix 3)**, or **(:fix 4)**.

Definition 2.8 (**Nominal equality formula).** A formula of the form $@_st$ where s and t are nominals is a nominal equality formula if it is an immediate conclusion of **(:nom ref)** or has such a formula as an ancestor on the branch while being an immediate conclusion of **(:nom)**

Definition 2.9 (**Root-subformula).** Where the root formula of a tableau Θ is written $\textit{root}_\Theta$, a formula $@_s\varphi$ occuring on a tableau Θ is called a *root-subformula* on Θ if it is a quasi-subformula of $\textit{root}_\Theta$.

Lemma 2.1 (**Subformula Property).** *Where Θ is a tableau branch in the FHTL calculus, any formula $@_s\varphi$ occuring on Θ is either a root-subformula, accessibility formula, nominal equality formula, or term equality formula.*

Proof. This is verified by checking the tableau rules. We observe the propositional, quantifier, and @ rules, along with **(:fix 1)**, **(:fix 2)**, and **(:fix 4)** can only take root-subformulae as premises and yield root-subformulae as conclusions. Definitionally **(ref)**, **(:3)**, and **(:func)** generate only term equality formulae and **(nom ref)** generates only nominal equality formulae. Whether the premise of **(:2)** is a nominal equality or root-subformula, the conclusion is a term equality. **(:1)** and **(:fix 3)** both preserve the "type" of their premise, since **(:1)**'s conclusion only changes the nominal prefix and if **(:fix 3)**'s premise is a root-subformula then its conclusion with its changed terms is still a quasi-subformula of the premise by definition. This leaves us with (F/P), ($\neg F/\neg P$), (F/P**–bridge**), **(sub)**, and **(nom)**. The premise of (F/P) must be either an accessibility formula or a root subformula, in the first case the second conclusion is a nominal equality, and in the second case the second conclusion is a root subformula, in either case the first conclusion is an accessibility formula. In the case of ($\neg F/\neg P$), the first premise can only be a root subformula since there's no way to introduce a negation, as a result the conclusion must be a root subformula. In the case of (F/P**–bridge**), the second premise is either an accessibility formula or a root subformula, and since the definition of quasi-subformula allows for substitution of (equivalent) nominals, the conclusion is the same kind of formula as the second premise. Likewise **(sub)**'s conclusion will be of the same type as its second premise since the definition of quasi-subformula allows for substitution of (equivalent) terms. And finally **(nom)**'s conclusion will be of the same type as its second premise since the definition of quasi-subformula allows for change of prefix nominals. □

Definition 2.10 (**$\prec_\Theta$).** 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 2.11 (**$\textit{Nom}_\Theta$).** The set of nominals and state variables which occur on Θ is written $\textit{Nom}_\Theta$

Lemma 2.2. *Where Θ is a tableau branch in the FHTL calculus, the graph $G = (\textit{Nom}_\Theta, \prec_\Theta)$ 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 $\textit{Nom}_\Theta$ under the relation $\prec_\Theta$ has a least element and each tree in G is wellfounded.

- *G is a tree*

  Every nominal in $\textit{Nom}_\Theta$ can be generated by at most one other nominal, and every nominal in $\textit{Nom}_\Theta$ 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 the F, P, $\neg H$, $\neg G$ rules to a premise of the form $@_i F\varphi$, $@_i P\varphi$, $@_i \neg H\varphi$, or $@_i \neg G\varphi$ 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 2.3. *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 Blackburn (2009) holds here as well.

□

Lemma 2.4 (Decreasing length). *Let Θ be a FHTL tableau branch, and s and t are nominals occuring on Θ. If $s \prec_\Theta t$ then $m_\Theta(s) > m_\Theta(t)$.*

Proof. Where φ is a formula of maximal length true at t on Θ, we need to show $m_\Theta(s) > |\varphi|$. Since $s \prec_\Theta t$ then φ must have been introduced to the branch by application of some *FHTL rule*. However it cannot have been introduced to the branch by any of the propositional rules, quantifier rules, @ rules, term rules, $\neg P$ or $\neg F$ since that would contradict the formula's maximality. □

Lemma 2.5 (Tableau construction termination). *Any tableau in the FHTL calculus is finite.*

Theorem 2.6. *The satisfiability of a finite set of FHTL sentences in a FHTL model is decidable.*

3 Model Checking via Tableaux

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 the smallest 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 and entities 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).

Consequently our use of tableaux for *FHTL* formulae will provide a decision procedure for their satisfiability within a finite model generated by some set of AMR sentences, rather than their general validity or invalidity, as is usually the case with tableaux methods. We develop an approach to using tableaux as a means of model checking for *FHTL* based on the approach in Bohn et al. (1998).

3.1 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:

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

4 AMR Interpretation in *FHTL*

4.1 Examples

(1) 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:
$@_{\textit{now}}\exists x[\texttt{form}(x) \wedge P\texttt{fill-out-03}(\texttt{Carl}, x)] \wedge @_{\textit{now}}\exists x[\texttt{form}(x) \wedge \forall y[\texttt{person}(y) \rightarrow F\texttt{submit-01}(y, x)]]$
d. Correct wrt plurality:
$@_{\textit{now}}\forall x[\texttt{form}(x) \wedge P\texttt{fill-out-03}(\texttt{Carl}, x)] \wedge @_{\textit{now}}\forall x[\texttt{form}(x) \wedge \forall y[\texttt{person}(y) \rightarrow F\texttt{submit-01}(y, x)]]$
e. Correct wrt reentrance (but not plurality) (maybe requires the passive for singular case?):
$@_{\textit{now}}\exists x[\texttt{form}(x) \wedge P\texttt{fill-out-03}(\texttt{Carl}, x) \wedge \forall y[\texttt{person}(y) \rightarrow F\texttt{submit-01}(y, x)]]$

(2) 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:
$@_{\textit{now}}\neg F\exists x[\texttt{car}(x) \wedge \neg P\texttt{notice-01}(x)]$
d. Technically correct:
$@_{\textit{now}}\neg F\exists x[\texttt{car}(x) \wedge \neg\forall y[\texttt{person}(y) \rightarrow P\texttt{notice-01}(x, y)]]$
e. Correct wrt particularity of the car:
$@_{\textit{now}}\neg F\neg\forall x[\texttt{person}(y) \rightarrow P\texttt{notice-01}(\texttt{car}, y)]]$

NB: Will complete these translations in full.

4.2 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.

4.3 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, {$\wedge, \vee, \rightarrow$})
        clauses = []
        for op ∈ 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)
    $\text{Arg}_0$, $\text{Arg}_1$ = GetArgs (AMR)
    return $@_{\text{nominal}}$ Apply ($\text{Arg}_0$, Apply ($\text{Arg}_0$, 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):
    if hasMods(UnaryPred) then
        mods = [name(UnaryPred)]
        for mod ∈ 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
```

References

Girish Shivanand Bhat. 1998. *Tableau-based approaches to model-checking*. Ph.D. thesis, North Carolina State University.

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 Patrick Blackburn. 2009. Terminating tableau calculi for hybrid logics extending k. *Electronic Notes in Theoretical Computer Science*, 231:21–39.

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.

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.

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.