Formalizing AMR Inference via Hybrid Logic Tableaux

CL Masters Thesis Defense

Eli Goldner

August 6, 2021

Introduction

- Semantic representation:
  - Capture meaning of natural language content.
  - Designed for software to manipulate and interpret.
- Abstract Meaning Representation (AMR):
  - Graph-based (DAG), nodes are *concepts*, edges are *relations*.
  - Built on predicative core of a sentence.
  - Ignores syntactic and fine grained semantic differences between sentences.
  - Concepts annotated with PropBank framed (entities, events, properties, states).

Introduction

(Core) AMR is reductionistic. Useful for:

- Annotation (esp. by non-experts).
- Semantic parsing (smaller target space).

Not useful for:

- Representing and recovering fine-grained meaning.
- Automating reasoning/inference.

Introduction

- AMR designers made a choice that works well in data-driven NLP.
- Modular AMR extensions.
- Some extensions afford richer logical interpretation:
- Automated inference for logics is a rich area with lots of tools.

Motivation

"Why do we need formal methods? Can't state-of-the-art language models do this already?"

- Statistically driven techniques are unnecessarily expensive for formal inference.
- Increasing need for ability to guarantee/verify properties of software:
  - Does the software give us the right *type* of result for an input?
  - Bias in NLP.

Approach

- Combine two AMR extensions for richer interpretation:
  - Scope and quantification (Pustejovsky et al., 2019)
  - Tense and aspect (Donatelli et al., 2018)
- Interpret these extended AMR into a logic that handles quantification and tense.
- Develop tableau methods for this logic:
  - General method for proving/disproving sentences in the logic.
  - Restricted method for checking if sentence holds in some model.

AMR with Scope and Quantification

- Disambiguates scope.
- Annotates central predicate and its arguments.
- Clearest path for AMR $\rightarrow$ standard first-order predicate logic.

AMR with Tense and Aspect

- Standard AMR structure.
- Central predicate annotated for:
  - Aspect.
  - Speech time.
  - Reference time.

Combined Extensions

- Assume each AMR has information from both extensions.
- Attach tense and aspect information to central predicate node.
- Extract a tense-sensitive FOPL representation (details later).

Modal Logic

- Propositional logic lets us form statements like $p \wedge (q \vee \neg r)$.
- Modal propositional logic extends propositional logic with an operator $\Diamond$, read as "possible". i.e. it is not possible that p and $\neg p$ are the case would be:

$$\neg\Diamond(p \wedge \neg p)$$

- (More) formal meaning of $\Diamond$: There is a *possible world* where p is true, and this possible world is *accesible* from the current one.
- The problem: the "current world" is an implicit notion dependent on context. Is there something more expressive?

Hybrid Logic

- Idea: take propositional modal logic, and add an operator @, that lets us know which world we're referring to.
- p or r is possible at world i: $@_i\Diamond(p \vee r)$
- In the above proposition i is called a nominal since it *names* some/is true at exactly one world.
- Everything true at the nominal j is true at the nominal i (they name the same world): $@_i j$

Hybrid Logic Variants

- Hybrid tense logic:
  - Two tense modalities: $\langle F \rangle$ and $\langle P \rangle$
  - World j is in the past of world i: $@_i Pj$
- Quantified hybrid logic:
  - Hybrid logic with first-order quantifiers, relation, and function symbols.
  - At n (now) there is a person for whom it is possible to own a car:

$$@_n(\exists x)(Person(x) \wedge (\exists y)(Car(y) \wedge \Diamond Afford(x, y)))$$

Hybrid Logic Variants

A problem

$$@_n(\exists x)(Person(x) \wedge (\exists y)(Car(y) \wedge \Diamond Afford(x, y)))$$

What's the domain of quantification?

- Option 1: The domain is the same at every world: *There is someone (out of all people all people at all worlds) who can afford some car (out of all cars at all worlds).*
- Option 2: Each world has a different domain: *There is someone (out of everyone in this world) who can afford some car (out of all cars in this world).*

The first option in quantified hybrid (or modal) logic is called possibilist quantification, the second is called actualist. When worlds are interpreted as times, the former is called eternalist quantification, and the latter is callled presentist. We will use presentist quantification.

First-Order Hybrid Tense Logic

First-order Hybrid Tense Logic (*FHTL*) is:

- Quantified hybrid logic, with tense modalities F and P instead of $\Diamond$.
- Presentist quantification:
  - Each world has its own domain.
  - Quantification is domain sensitive.

First-Order Hybrid Tense Logic

In logic models are interpretations of symbols and constants in the language. An *FHTL* model $\mathfrak{M}$ is a tuple

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

Where:

- T is a set of times/worlds.
- $\mathcal{R}$ is the binary accesibility relation over times.
- D_t is the domain of a time t
- I_{nom} assigns nominals to worlds.
- I_t interprets the value of terms at a time t.

The satisfiability of a formula with free variables depends on a variable assignment function. A formula that does not depend on variable assignment (every variable is bound by a quantifier) is called a *sentence*.

First-Order Hybrid Tense Logic

- Two main tasks for a sentence $@_s\varphi$:
  - Theorem proving – is $@_s\varphi$ true regardless of the model?
  - Model checking – is $@_s\varphi$ true in a given model $\mathfrak{M}$?
- For both we use versions of the tableau method.

Tableau Method

- Tableau for a formula is a tree structure.
- A tableau caclulus for a logic breaks down and transforms formulae
- Rules of the calculus use semantics of connectives/quantifiers/modifiers.
- *FHTL* tableau based on *QHL* tableau modified for tense rules.

Tableau Method

- A tableau branch is a subtree of the main tableau tree.
- A tableau *branches* for rules involving disjunctions.
- A branch is *closed* if it contains both a formula $@_s\varphi$ and its negation $@_s\neg\varphi$. Otherwise it is open.
- A branch is *saturated* if no rules of the calculus can be applied without adding a redundant formula on the branch.

Tableau Example Rules

$$\frac{@_s F\varphi}{@_s Fa}\ (F)^{12}$$

$$@_a\varphi$$

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

$$\frac{@_s(\exists x)\varphi}{@_s\varphi[s{:}p/x]}\ (\exists)^{3}$$

$$\frac{@_i j{:}t = k{:}s \qquad @_i\varphi}{@_i\varphi[j{:}t//k{:}s]}\ \textbf{(sub)}^{4}$$

[1] The nominal a is new to the branch.

[2] The formula φ is not a nominal.

[3] $s{:}p$ is new to the branch.

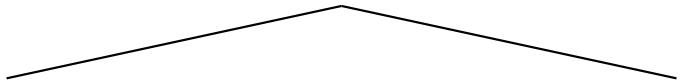
[4] $\varphi[j{:}t//k{:}s]$ is φ where some occurences of $j{:}t$ have been replaced by $k{:}s$.

FHTL Tableau Example

(1)

(2) $@_s(\exists x)[P((\exists y)[f(x,y) = f(y,x)]) \vee \neg(\exists z)[x = z]]$

(3) $@_s P((\exists y)[f(s_1,y) = f(y,s_1)]) \vee \neg(\exists z)[s_1 = z]$



(4)	$@_s P((\exists y)[f(s_1,y) = f(y,s_1)])$	$@_s \neg(\exists z)[s_1 = y]$
(5)	$@_s Pt$	$@_s \neg[s_1 = s_1]$
(6)	$@_t(\exists y)[f(s_1,y) = f(y,s_1)]$	$@_s[s_1 = s_1]$
(7)	$\ldots$	$\otimes$

Tableau Proofs

- The *root formula* of a tableau is unsatisfiable if every branch of the tableau closes.
- If we want to prove $@_s\varphi$ then we begin the tableau with $@_s\neg\varphi$ (proof by contradiction).
- Need to show for tableau method (or any proof system) that it is:
  - Sound – if a tableau is closed then the root formula is unsatisfiable.
  - Complete – if a formula is unsatisfiable it has a closed tableau proof.

Tableau Proofs

- Soundness is demonstrated by checking the rules.
- Completeness is demonstated by contrapositive.
- Open tableau branch $\rightarrow$ there is a model which satisfies the root.
- Construct the model out of equivalence classes of everything that shows up on the branch.

Tableaux for Model Checking

- So far the tableau method has been about general theorem proving.
- Problem: no tableau construction method guaranteed to terminate (if there was we'd have decidability of first-order logic).
- Most of the time however we are reasoning about an AMR/*FHTL* sentence in some local context.
- Revise tableau rules, trade completeness for termination.

Tableaux for Model Checking

NB: Terms in *FHTL* have nominal prefixes since their value can depent on the world they are evaluated in, i.e. $i{:}t$, $i{:}j{:}t$. There are three things that prevent tableau construction from terminating:

- Universal tableau rules ($\neg\exists$, $\forall$) – can generate an arbitrary number of conclusions.
- Term rules – can keep adding prefixes to terms.
- Nominal rules – generating redundant nominals (our complete rules take care of this).

- Idea: give tableau a reasonable chance to close, but make sure it terminates.
- Fix some $q \in \mathbb{N}^+$ and let universal rules generate at most q conclusions. (Fitting (1988) says $q = 1$ works surprisingly often.)
- Restrict term rules to depend on on nominals and terms already on the branch. Prevent redundant prefixes (no $i{:}j{:}i{:}t$ etc.).

Model checking procedure

- Build tableau using restricted rules.
- Check tableau tree from leaves up, skipping closed branches.
- Every formula on the branch has a set of variable and parameter assignments which satisfy it in the model.
- Solve for a formula's set based on the sets of its conclusions.
- If the root formula is satisfied in the model by every variable assignment then it holds.

Model Checking Example

- *Every desk will have a computer located there.*
- AMR with quantification and tense:

```
(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)
```

- *FHTL* translation:
  $@_{now}(\forall y)[\text{desk}(y) \rightarrow (\exists x)[\text{computer}(x) \wedge F(\text{be-located-at-91}(x, y))]]$

Model Checking Example

Define a small *FHTL* model $\mathfrak{M} = (T, \mathcal{R}, (D_t)_{t \in T}, I_{nom}, (I_t)_{t \in T})$ where:

$$T = \{yesterday, now, tomorrow\}$$

$$\mathcal{R} = \{(yesterday, now), (now, tomorrow), (yesterday, tomorrow)\}$$

$$I_{nom} = \{(y, yesterday), (n, now), (t, tomorrow)\}$$

$$I_{yesterday}(\texttt{be-located-at-91}) = \ldots$$

$$I_{now}(\texttt{be-located-at-91}) = \ldots$$

$$I_{tomorrow}(\texttt{be-located-at-91}) = \{\langle computer_2, desk_2\rangle, \langle computer_1, desk_1\rangle\}$$

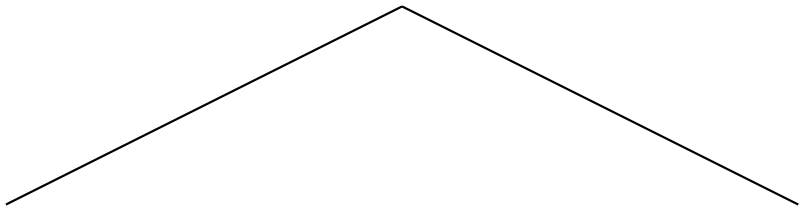
$$D_{yesterday} = \{computer_1, desk_1\}$$

$$D_{now} = \{computer_1, computer_2, desk_1, desk_2, desk_3\}$$

$$D_{tomorrow} = \{computer_1, computer_2, desk_1, desk_2\}$$

Model Checking Example

$@_n(\forall y)[\text{desk}(y) \rightarrow (\exists x)[\text{computer}(x) \wedge F(\text{located}(x, y))]]$
$@_n\text{desk}(n{:}p_1) \rightarrow (\exists x)[\text{computer}(x) \wedge F(\text{located}(x, n{:}p_1))]$



$@_n\neg\text{desk}(n{:}p_1)$

$@_n(\exists x)[\text{computer}(x) \wedge F(\text{located}(x, n{:}p_1))]$
$@_n\text{computer}(n{:}p_2) \wedge F(\text{located}(n{:}p_2, n{:}p_1))$
$@_n\text{computer}(n{:}p_2)$
$@_nF(\text{located}(n{:}p_2, n{:}p_1))$
$@_nFs$
$@_s\text{located}(n{:}p_2, n{:}p_1)$

Where we assign $s = t = \mathit{tomorrow}$, $n{:}p_2 = \mathit{computer}_1$ and $n{:}p_1 = \mathit{desk}_2$, or $n{:}p_2 = \mathit{computer}_2$ and $n{:}p_1 = \mathit{desk}_1$ we see that $\mathfrak{M}$ satisfies the *FHTL* sentence.

Extraction

In this new annotation time information is of the form:
`:<aspect> <polarity> :time (<reference> :op1 (<speech>))`

- `<speech>` is some $t \in T$, so formula has the prefix $@_t\varphi$
- `<reference>` (if present) is the tense, e.g. $@_t P\psi$
- We don't make use of aspect (right now).

Interpretation

Every AMR in this annotation is either of the form (for a connective `or`, `and`, or `cond`):

```
(<connective>
    :op1 (...)
    :op2 (...)
    :op3 (...))
```

Or

```
(s / scope
       :pred (...
          :ARG0 (...)
          :ARG1 (...))
       :ARG0 <primary-scope>
       :ARG1 <secondary-scope>)
```

Interpretation Examples

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

$$[\![\texttt{op1}]\!] \vee [\![\texttt{op2}]\!] \vee [\![\texttt{op3}]\!]$$

```
(s / scope
      :pred (...
         :ARG0 (...)
         :ARG1 (...))
      :ARG0 <primary-scope>
      :ARG1 <secondary-scope>)
```

$$[\![\texttt{<primary-scope>}]\!]([\![\texttt{<secondary-scope>}]\!]([\![\texttt{pred}]\!]))$$

Interpretation Examples

```
(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)
```

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

$$[\![\texttt{now}]\!] \rhd \mathit{now}$$

$$[\![\texttt{after}]\!] \rhd F$$

```
(c / computer)
```

$$\lambda\varphi.(\exists x)[\mathit{Computer}(x) \wedge \varphi(x)]$$

```
(d / desk
     :quant (e/ every))
```

$$\lambda\varphi.(\forall x)[\mathit{Desk}(x) \rightarrow \varphi(x)]$$

Interpretation Example

```
(:pred (b / be-located-at-91 :ongoing -
                             :complete +
                             :time (a / after
                                      :op1 (n / now))
```

$$\lambda\varphi.\lambda x.\lambda y.\varphi(\texttt{be-located-at-91}(x, y))$$

Interpretation Example

We'll just do the first application.

$$\lambda\varphi.(\lambda\varphi'.(\exists c)(\mathit{computer}(c)\wedge\varphi'(c)))(\lambda\psi.(\lambda\psi'.\psi'(\lambda x.\lambda y.\mathit{loc}(x,y)))(\lambda\gamma.\varphi(\gamma(\psi)))\rhd_\beta$$

$$\lambda\varphi.(\lambda\varphi'.(\exists c)(\mathit{computer}(c)\wedge((\lambda\psi'.\psi'(\lambda x.\lambda y.\mathit{loc}(x,y)))(\lambda\gamma.\varphi(\gamma(c)))))\rhd_\beta$$

$$\lambda\varphi.(\lambda\varphi'.(\exists c)(\mathit{computer}(c)\wedge(\varphi(\lambda y.\mathit{loc}(c,y)))))$$

Conclusion

- We explored why AMR can benefit from formal inference.
- Described a combination of AMR extensions.
- Sketched a first-order hybrid logic variant with general proof procedure and model checking methods.
- Demonstrated how sentences are be handled by these methods.
- Showed how AMR sentences can be translated into *FHTL*

References I

Carlos Areces, Diego Figueira, Daniel Gorín, and Sergio Mera. 2009. Tableaux and model checking for memory logics. In *International conference on automated reasoning with analytic tableaux and related methods*, pages 47–61. Springer.

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

Thomas Bolander and Patrick Blackburn. 2007. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554.

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

References II

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.

Hardi Hungar, Orna Grumberg, and Werner Damm. 1995. What if model checking must be truly symbolic. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pages 1–20. Springer.

References III

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.