Hybrid Logic in the Isabelle Proof Assistant: Benefits, Challenges and the Road Ahead

Asta Halkjær From

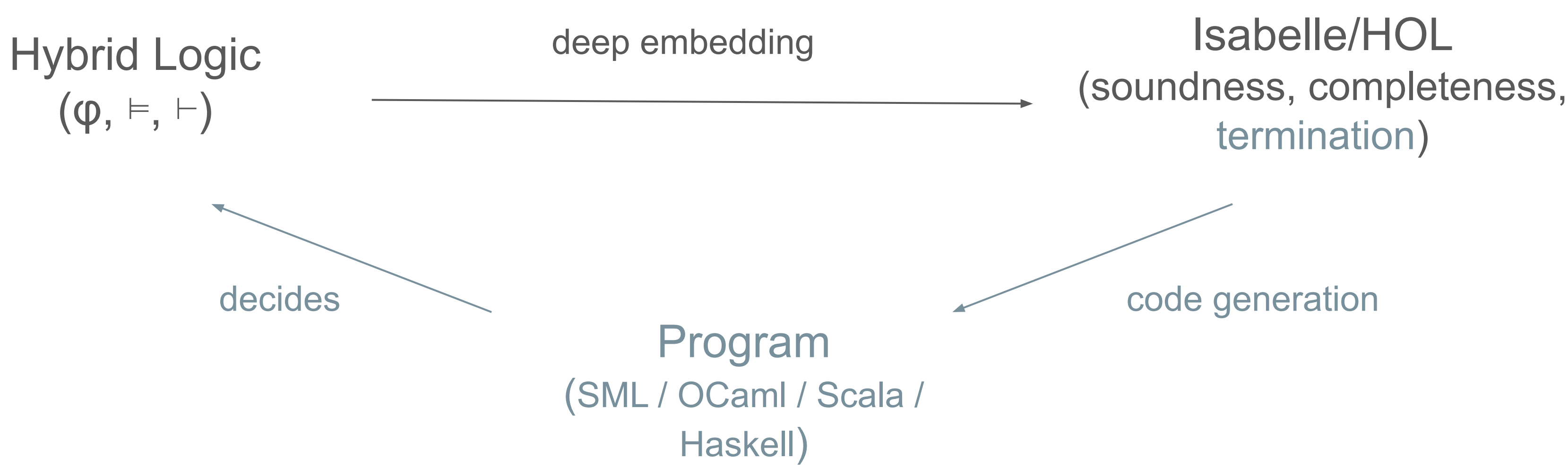
Technical University of Denmark

AiML 2020, August 26.

Introduction

Case study: Tableau system ST^A for basic hybrid logic with a formalization in Isabelle/HOL (4900+ lines in the Archive of Formal Proofs): https://devel.isa-afp.org/entries/Hybrid_Logic.html

Based on ST* by Blackburn, Bolander, Braüner and Jørgensen.



Hybrid Logic

Modal logic with names for worlds (nominals) and satisfaction operators ($@_i$).

$$\phi, \psi ::= x \mid i \mid \neg\phi \mid \phi \vee \psi \mid \Diamond\phi \mid @_i\phi$$

Semantics based on Kripke models ((*W*, *R*), *V*) and an assignment *g*.

W: underlying set, *R*: binary relation, *V*: unary relation, *g*: map from nominals to worlds.

$$\begin{array}{lll}
\mathfrak{M}, g, w \models x & \text{iff} & w \in V(x) \\
\mathfrak{M}, g, w \models i & \text{iff} & g(i) = w \\
\mathfrak{M}, g, w \models \neg\phi & \text{iff} & \mathfrak{M}, g, w \not\models \phi \\
\mathfrak{M}, g, w \models \phi \vee \psi & \text{iff} & \mathfrak{M}, g, w \models \phi \text{ or } \mathfrak{M}, g, w \models \psi \\
\mathfrak{M}, g, w \models \Diamond\phi & \text{iff} & \text{for some } w',\ wRw' \text{ and } \mathfrak{M}, g, w' \models \phi \\
\mathfrak{M}, g, w \models @_i\phi & \text{iff} & \mathfrak{M}, g, g(i) \models \phi
\end{array}$$

Hybrid Logic in Isabelle – Syntax

The syntax of our object logic becomes a **datatype** in the metalogic:

```
datatype ('a, 'b) fm
  = Pro 'a
  | Nom 'b
  | Neg ‹('a, 'b) fm› (‹¬ _› [40] 40)
  | Dis ‹('a, 'b) fm› ‹('a, 'b) fm› (infixr ‹∨› 30)
  | Dia ‹('a, 'b) fm› (‹◇ _› 10)
  | Sat 'b ‹('a, 'b) fm› (‹@ _ _› 10)
```

Values of type *('a, 'b) fm* are built from the above constructors.
Propositional symbols are drawn from type variable 'a and nominals from 'b.

We specify the usual infix notation in bold.

Hybrid Logic in Isabelle – Semantics

Type variable ‘w represents the set of worlds:

```
datatype ('w, 'a) model =
  Model (R: ‹'w ⇒ 'w set›) (V: ‹'w ⇒ 'a ⇒ bool›)
```

The semantics interprets the object logic in the metalogic (HOL, not English).

```
primrec semantics
  :: ‹('w, 'a) model ⇒ ('b ⇒ 'w) ⇒ 'w ⇒ ('a, 'b) fm ⇒ bool›
  (‹_, _, _ ⊨ _› [50, 50, 50] 50) where
  ‹(M, _, w ⊨ Pro x) = V M w x›
| ‹(_, g, w ⊨ Nom i) = (w = g i)›
| ‹(M, g, w ⊨ ¬ p) = (¬ M, g, w ⊨ p)›
| ‹(M, g, w ⊨ (p ∨ q)) = ((M, g, w ⊨ p) ∨ (M, g, w ⊨ q))›
| ‹(M, g, w ⊨ ◇ p) = (∃v ∈ R M w. M, g, v ⊨ p)›
| ‹(M, g, _ ⊨ @ i p) = (M, g, g i ⊨ p)›
```

Seligman-Style Tableau System – Blocks

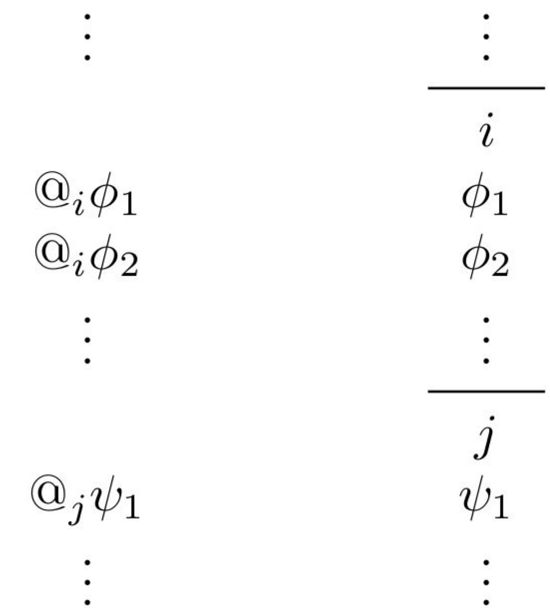
Formulas are true relative to a given world.

- Internalized calculus: Satisfaction statements only.
- Seligman-style: Group formulas into blocks named by *opening nominals*.

Blocks and branches are modelled naturally:

```
type_synonym ('a, 'b) block = ‹('a, 'b) fm list × 'b›
type_synonym ('a, 'b) branch = ‹('a, 'b) block list›
```

We know exactly what a branch is formally.



(a) Internalized. (b) Seligman-style.

Seligman-Style Tableau System – Rules

Horizontal lines indicate that input can appear on an earlier block.

Vertical lines indicate extensions / justifications.

Above each input formula is the nominal of its block.

Multiple inputs are written next to each other.

The set *A* is fixed in advance (e.g. the root nominals)

$$\dfrac{\begin{array}{c} a \\ \phi \lor \psi \end{array}}{\begin{array}{c} a \\ / \quad \backslash \\ \phi \qquad \psi \end{array}} \quad (\lor)$$

$$\dfrac{\begin{array}{c} a \\ \neg(\phi \lor \psi) \end{array}}{\begin{array}{c} a \\ | \\ \neg\phi \\ \neg\psi \end{array}} \quad (\neg\lor)$$

$$\dfrac{\begin{array}{c} a \\ \neg\neg\phi \end{array}}{\begin{array}{c} a \\ | \\ \phi \end{array}} \quad (\neg\neg)$$

$$\dfrac{\begin{array}{c} a \\ \Diamond\phi \end{array}}{\begin{array}{c} a \\ | \\ \Diamond i \\ @_i\phi \end{array}} \quad (\Diamond)^1$$

$$\dfrac{\begin{array}{cc} a & a \\ \neg\Diamond\phi & \Diamond i \end{array}}{\begin{array}{c} a \\ | \\ \neg @_i\phi \end{array}} \quad (\neg\Diamond)$$

$$\dfrac{|}{i} \quad \text{GoTo}^2$$

$$\dfrac{\begin{array}{cc} b & b \\ a & \phi \end{array}}{\begin{array}{c} a \\ | \\ \phi \end{array}} \quad \text{Nom}^3$$

$$\dfrac{\begin{array}{cc} i & i \\ \phi & \neg\phi \end{array}}{\begin{array}{c} a \\ | \\ \times \end{array}} \quad \text{Closing}$$

$$\dfrac{\begin{array}{c} b \\ @_a\phi \end{array}}{\begin{array}{c} a \\ | \\ \phi \end{array}} \quad (@)$$

$$\dfrac{\begin{array}{c} b \\ \neg @_a\phi \end{array}}{\begin{array}{c} a \\ | \\ \neg\phi \end{array}} \quad (\neg @)$$

[1] i is fresh, $i \notin A$ and ϕ is not a nominal.
[2] i is not fresh.
[3] If $\phi = i$ or $\phi = \Diamond i$ for some nominal i then $i \in A$.

Seligman-Style Tableau System – Restricting GoTo

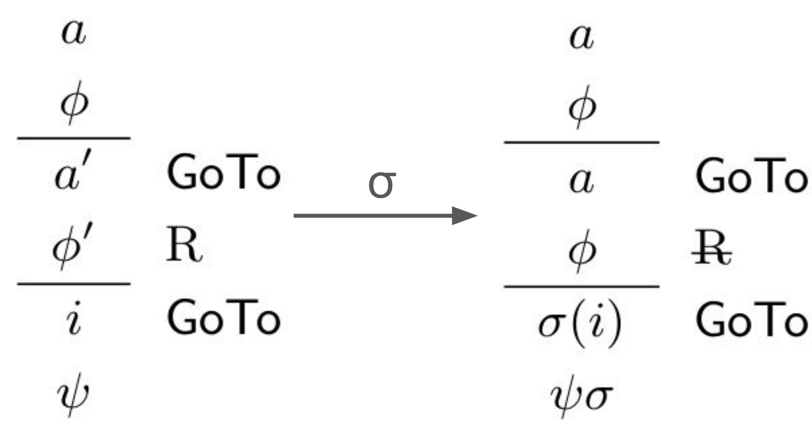
Rule out repeated applications of GoTo: *GoTo cannot be applied twice in a row.*

Easy to respect *informally* when proving e.g. substitution lemma:
if Θ has a closing tableau then so does Θσ.

Induction in Isabelle “mimics” the construction of the closing tableau for Θ but this may be a problem:

We have to *handle detours explicitly*:

- New induction principle
- Substitution + *pruning*
- Weaker lemma
- *Different restriction*



Seligman-Style Tableau System – Potential

Handle detours by assuming more starting "potential":

R4 The GoTo rule consumes one *potential*. The remaining rules add one unit of potential and we are allowed to start from any amount.

Allows for *local detours*.

Still prevents repeated GoTo.

Easy to formalize.

0.	a		
1.	$\neg(\neg @_i\phi \vee @_i\phi)$		[0]
2.	$\neg\neg @_i\phi$	$(\neg\vee)$ 1	[1]
3.	$\neg @_i\phi$	$(\neg\vee)$ 1	[2]
4.	$@_i\phi$	$(\neg\neg)$ 2	[3]
5.	i	GoTo	[2]
6.	$\neg\phi$	$(\neg @)$ 3	[3]
7.	ϕ	$(@)$ 4	[4]
	×		

Seligman-Style Tableau System – Formalization

The turnstile predicate holds if the branch can be closed (wrt. *A* and *n*). E.g.

```
| Neg:
  ‹(¬ ¬ p) at a in (ps, a) # branch ⟹
   new p a ((ps, a) # branch) ⟹
   A, Suc n ⊢ (p # ps, a) # branch ⟹
   A, n ⊢ (ps, a) # branch›
```

We can then machine verify results like soundness and completeness:

```
theorem soundness_fresh:
  assumes ‹A, n ⊢ [([¬ p], i)]› ‹i ∉ nominals p›
  shows ‹M, g, w ⊨ p›

theorem completeness:
  fixes p :: ‹('a :: countable, 'b :: countable) fm›
  assumes
    inf: ‹infinite (UNIV :: 'b set)› and
    valid: ‹∀(M :: ('b set, 'a) model) g w. M, g, w ⊨ p›
  shows ‹nominals p, 1 ⊢ [([¬ p], i)]›
```

Formalizing Termination

We formalized an *inductive* predicate $\vdash$: does a *finite* branch close?
Now we need a *coinductive* well-formedness predicate on finitely branching, possibly *infinite* trees:

```
codatatype (labels: 'a) tree = Node (getLabel: 'a) (getSubs: ‹'a tree list›)
```

That is, a codatatype may have *~~turtles~~ constructors all the way down.*

Current work:

```
theorem ‹twf A tab ⟹ fin tab›
  oops
```

twf (coinductive): all rules are applied correctly.
fin (inductive): the tableau is finite.

Concluding Remarks

“PROOF. This is easily seen by going through each of the tableau rules.”
Not always as easy in Isabelle!

But! We *can* do our metatheory in higher-order logic instead of natural language.
No omissions, no ambiguity.

Verify the definitions ⇒ trust the results.

Future work:

- Formalize termination.
- Produce an executable decision procedure.

References

Seligman, J. (2001). Internalization: The case of hybrid logics. *Journal of logic and computation*, *11*(5), 671-689.

Blackburn, P., Bolander, T., Braüner, T., & Jørgensen, K. F. (2017). Completeness and termination for a Seligman-style tableau system. *Journal of Logic and Computation*, *27*(1), 81-107.

Jørgensen, K. F., Blackburn, P., Bolander, T., & Braüner, T. (2016). Synthetic completeness proofs for Seligman-style tableau systems. *Advances in Modal Logic*, *11*, 302-321.

From, A. H., Blackburn, P., & Villadsen, J. (2020, July). Formalizing a Seligman-Style Tableau System for Hybrid Logic. In *International Joint Conference on Automated Reasoning* (pp. 474-481). Springer, Cham.

Bolander, T., & Blackburn, P. (2007). Termination for hybrid tableaus. *Journal of Logic and Computation*, *17*(3), 517-554.

Blanchette, J. C., Popescu, A., & Traytel, D. (2017). Soundness and completeness proofs by coinductive methods. *Journal of Automated Reasoning*, *58*(1), 149-179.