Formalizing a Seligman-Style Tableau System for Hybrid Logic

Asta Halkjær From[1], Patrick Blackburn[2] and Jørgen Villadsen[1]

[1] Technical University of Denmark, [2] Roskilde University

IJCAR 2020, July 1.

Introduction

We present a 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

The system is based on ST* by Blackburn, Bolander, Braüner and Jørgensen.

Our short paper described work in progress and now the work has progressed. We show the latest system here.

The focus is on the process and results, not the proofs.

See the paper for details, related work and bibliography.

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*: maps nominals to worlds.

$$\mathfrak{M}, g, w \models x \quad \text{iff} \quad w \in V(x)$$

$$\mathfrak{M}, g, w \models i \quad \text{iff} \quad g(i) = w$$

a nominal only holds at the world it names

$$\mathfrak{M}, g, w \models \neg\phi \quad \text{iff} \quad \mathfrak{M}, g, w \not\models \phi$$

$$\mathfrak{M}, g, w \models \phi \vee \psi \quad \text{iff} \quad \mathfrak{M}, g, w \models \phi \text{ or } \mathfrak{M}, g, w \models \psi$$

$$\mathfrak{M}, g, w \models \Diamond\phi \quad \text{iff} \quad \text{for some } w', \ wRw' \text{ and } \mathfrak{M}, g, w' \models \phi$$

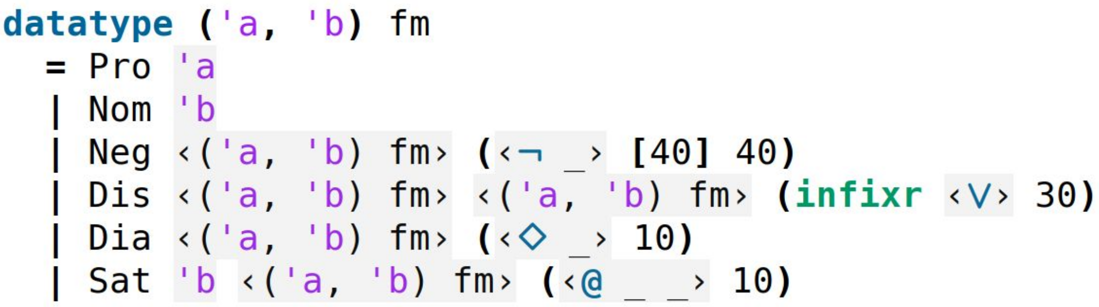
$$\mathfrak{M}, g, w \models @_i\phi \quad \text{iff} \quad \mathfrak{M}, g, g(i) \models \phi$$

satisfactions statements shift perspective

Hybrid Logic in Isabelle – Syntax

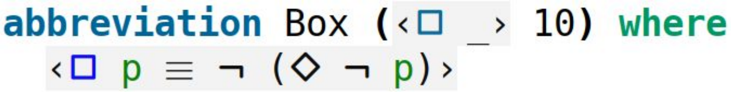
Deep embedding of the logic.

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)
```

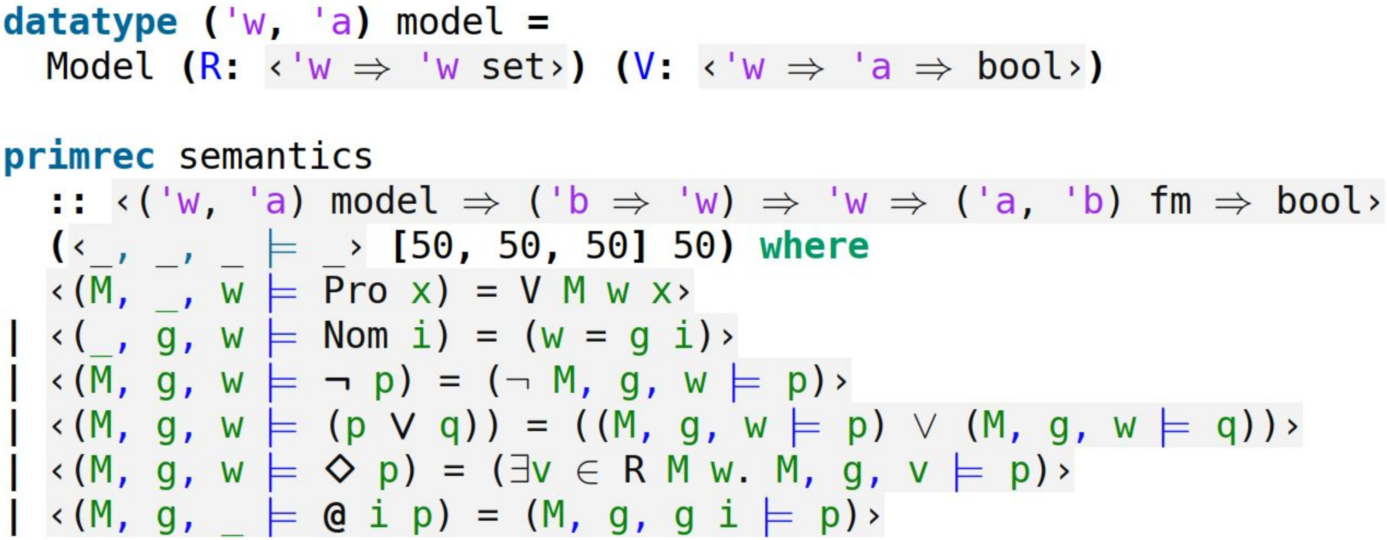
We specify the usual infix notation and can give other operators as abbreviations:



```
abbreviation Box (‹□ _› 10) where
  ‹□ p ≡ ¬ (◇ ¬ p)›
```

Hybrid Logic in Isabelle – Semantics

Type variables represent the universes of worlds (‘w), propositional symbols (‘a) and nominals (‘b).



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

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)›
```

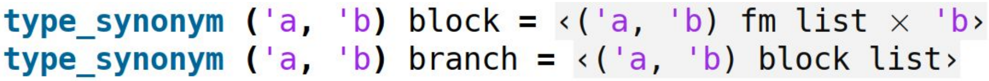
We can now talk about the logic in the formal language of higher-order logic.

Seligman-Style Tableau System – Blocks

Formulas are true relative to a given world.

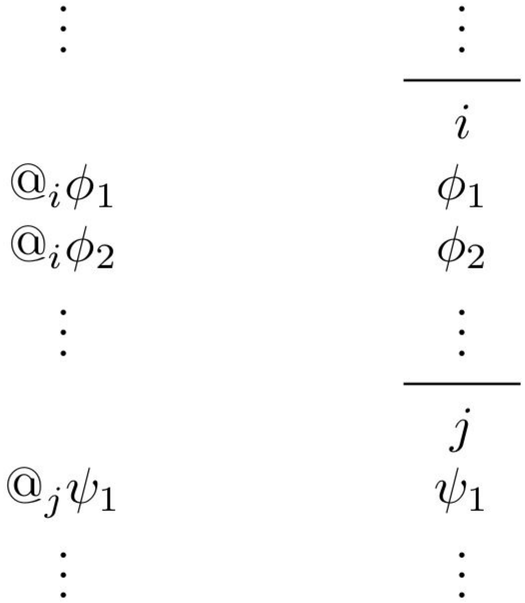
- Internalized calculus: Satisfaction statements only.
- Seligman-style: Group formulas into blocks named by *opening nominals*.

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 assume that the initial block is always named to obtain this simple modelling.



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

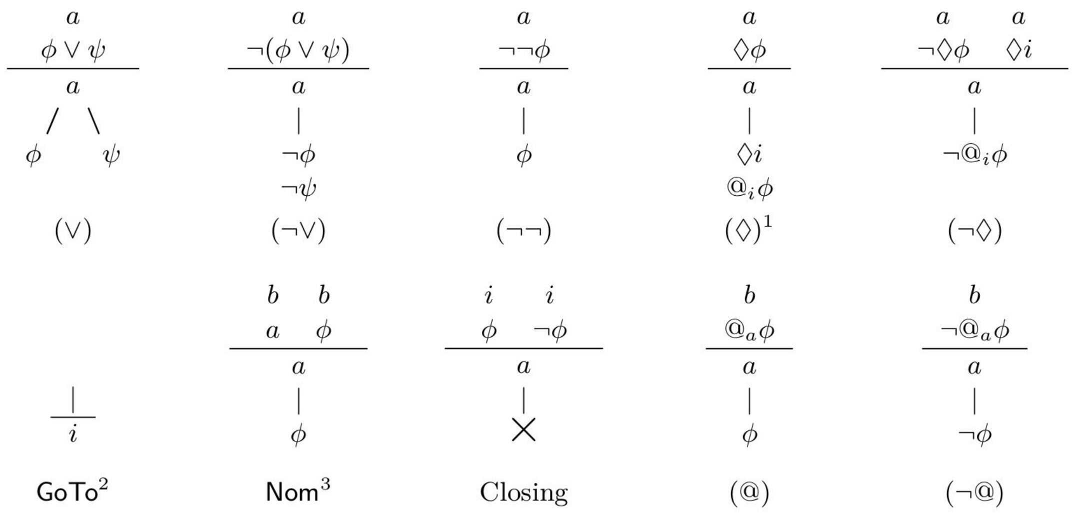
Seligman-Style Tableau System – Rules

Beware: The horizontal line is the block separator.

The vertical lines give extensions / justifications.

Each input formula must appear on a block named by the nominal written above it.

Multiple inputs are written next to each other.



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

$(\lor)$

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

$(\neg\lor)$

$$\begin{array}{c} a \\ \neg\neg\phi \\ \hline a \\ | \\ \phi \end{array}$$

$(\neg\neg)$

$$\begin{array}{c} a \\ \Diamond\phi \\ \hline a \\ | \\ \Diamond i \\ @_i\phi \end{array}$$

$(\Diamond)$[1]

$$\begin{array}{cc} a & a \\ \neg\Diamond\phi & \Diamond i \\ \hline \multicolumn{2}{c}{a} \\ \multicolumn{2}{c}{|} \\ \multicolumn{2}{c}{\neg @_i\phi} \end{array}$$

$(\neg\Diamond)$

$$\begin{array}{c} | \\ \hline i \end{array}$$

GoTo[2]

$$\begin{array}{cc} b & b \\ a & \phi \\ \hline \multicolumn{2}{c}{a} \\ \multicolumn{2}{c}{|} \\ \multicolumn{2}{c}{\phi} \end{array}$$

Nom[3]

$$\begin{array}{cc} i & i \\ \phi & \neg\phi \\ \hline \multicolumn{2}{c}{a} \\ \multicolumn{2}{c}{|} \\ \multicolumn{2}{c}{\times} \end{array}$$

Closing

$$\begin{array}{c} b \\ @_a\phi \\ \hline a \\ | \\ \phi \end{array}$$

$(@)$

$$\begin{array}{c} b \\ \neg @_a\phi \\ \hline a \\ | \\ \neg\phi \end{array}$$

$(\neg @)$

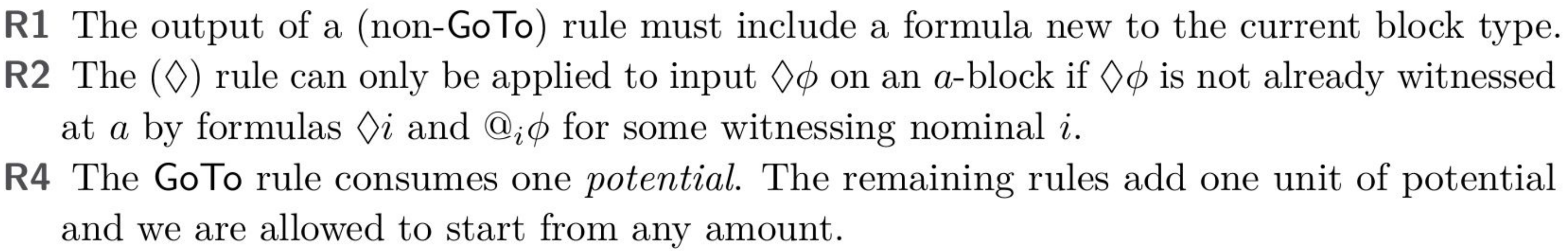
[1] i is fresh, $i \notin A$ and ϕ is not a nominal.

[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 – Restrictions

We want to rule out the construction of infinite branches.



R1 The output of a (non-GoTo) rule must include a formula new to the current block type.

R2 The ($\Diamond$) rule can only be applied to input $\Diamond\phi$ on an a-block if $\Diamond\phi$ is not already witnessed at a by formulas $\Diamond i$ and $@_i\phi$ for some witnessing nominal i.

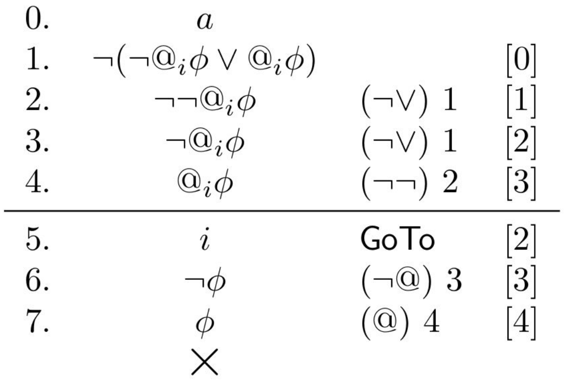
R4 The GoTo rule consumes one *potential*. The remaining rules add one unit of potential and we are allowed to start from any amount.

Our @-rules adhere to the **R5** given by Blackburn et al.

R1 and **R2** can be *lifted*.

Potential constrains GoTo without hindering induction.

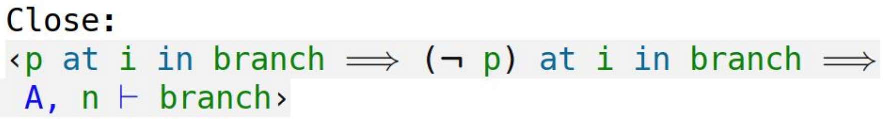
We restrict the Nom rule by a set of allowed *nominals.*



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]
	×		

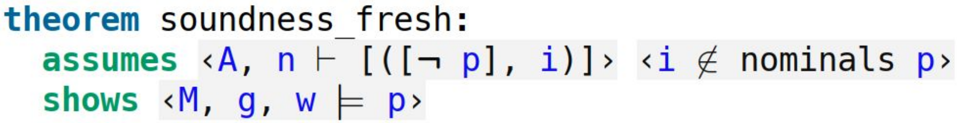
Seligman-Style Tableau System – Formalization

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



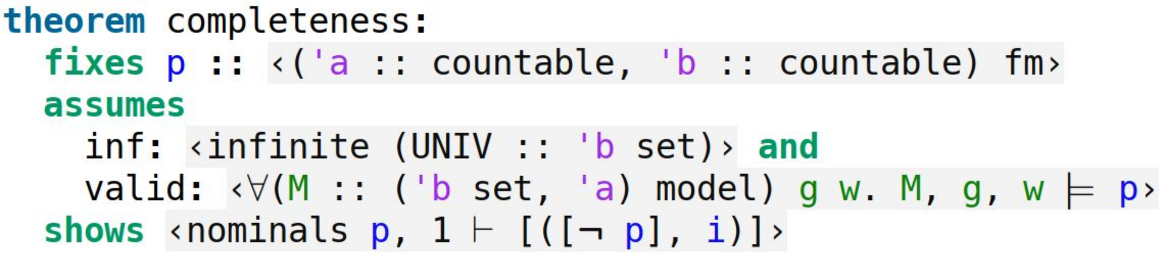
```
Close:
‹p at i in branch ⟹ (¬ p) at i in branch ⟹
 A, n ⊢ branch›
```

We can then machine verify results like soundness:



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

And completeness:



```
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)]›
```

Concluding Remarks

The nominals of hybrid logic allow us to witness the diamond modality.

Seligman-style rules work on arbitrary formulas due to explicit perspective shifts.

We can model the logic and proof system in Isabelle/HOL in a natural way.

In doing so, we gain absolute trust in our results: no omissions, no ambiguity.

Next steps:

- Show that the system is terminating.
- Verify a decision procedure based on the calculus.

I have an upcoming short presentation at Advances in Modal Logic 2020.