Hybrid Logic

MSc Defense, Asta Halkjær From

Formalizing a Seligman-Style Tableau System

Preface

- MSc in Computer Science and Engineering, Technical University of Denmark.
- Thesis period: 19 August 2019 to 19 January 2020 (30 ECTS).
- Defense: 29 January 2020.
- Supervisors:
  - Jørgen Villadsen
  - Alexander Birch Jensen (co-supervisor)
  - Patrick Blackburn (external supervisor, Roskilde University)

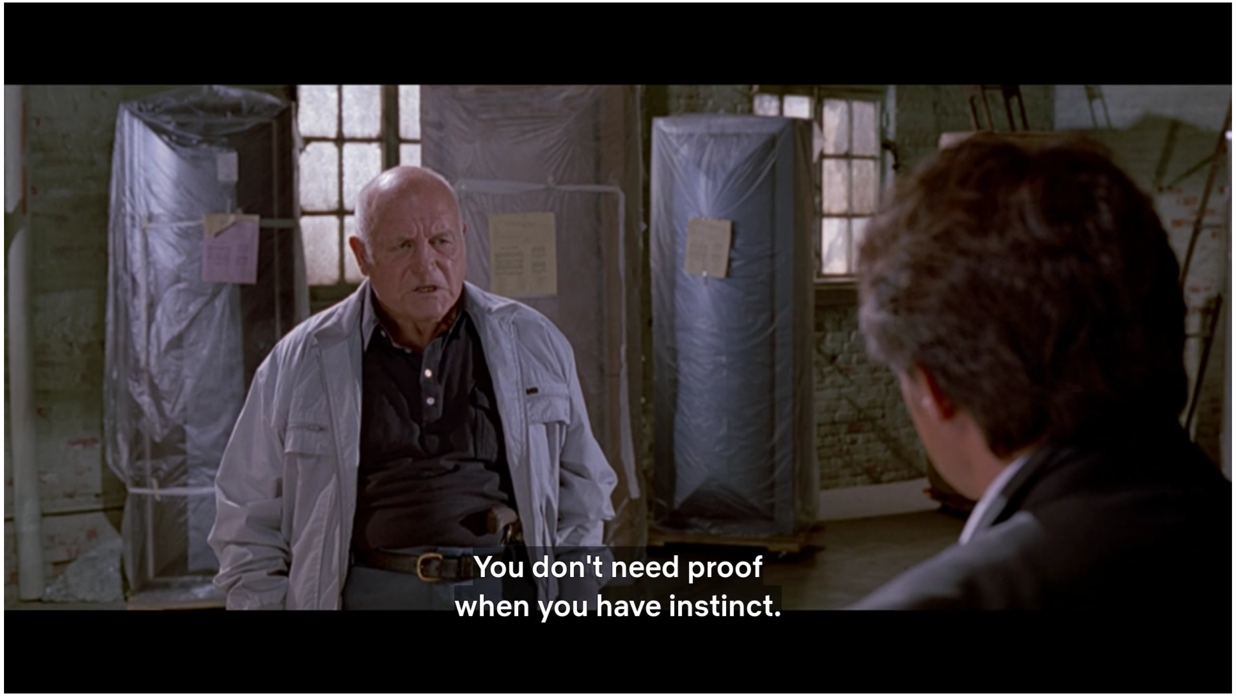
Overview

- Isabelle
  - Archive of Formal Proofs
- Hybrid Logic
- Seligman-Style Tableau System
- Restrictions Towards Termination
- Lifting Restrictions
- Admissible Bridge
- Completeness
  - Hintikka definition
- Future Work
- Conclusion

Isabelle/HOL Proof Assistant

- Generic proof assistant Isabelle
  - Isabelle/HOL is the higher-order logic instance.
- Express mathematical statements and proofs in a formal language
  - Unambiguous definitions.
  - Machine-checked proofs.
  - Proof search (and proof search search).
  - Counterexample search.
- LCF architecture
  - Abstract type of theorems.
  - Trusted kernel.

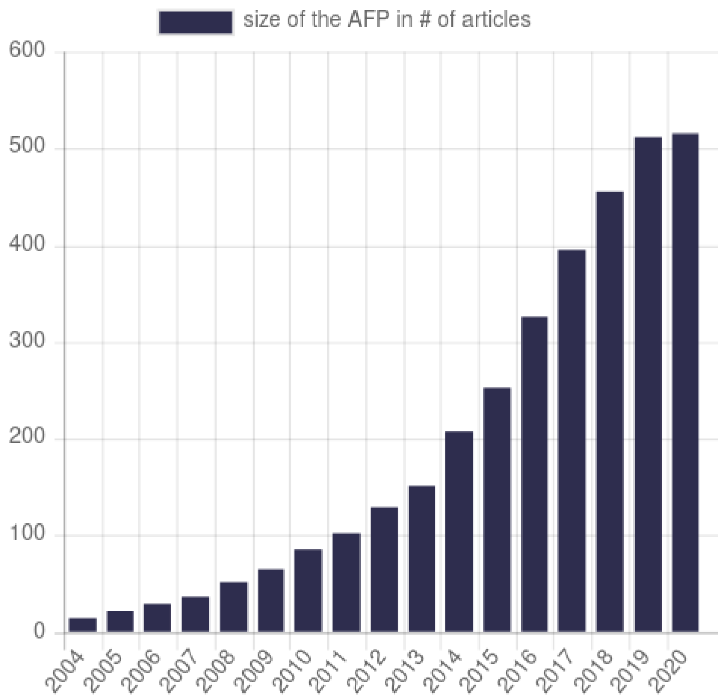
Mood



Isabelle

Archive of Formal Proofs I

- “[C]ollection of proof libraries, examples, and larger scientific developments, mechanically checked in the theorem prover Isabelle.”
- Refereed submissions.
- Formalizations kept up to date with Isabelle.
- Statistics
  - Number of Articles: 516
  - Number of Authors: 340
  - Number of lemmas: ~141,100
  - Lines of Code: ~2,452,800
  - https://www.isa-afp.org/statistics.html



Archive of Formal Proofs II (index by topic)

Computer Science (293)

- Automata and Formal Languages (39)
- Algorithms (75)
- Concurrency (19)
- Data Structures (50)
- Functional Programming (21)
- Games (1)
- Hardware (1)
- Networks (6)
- Programming Languages (83)
- Security (39)
- Semantics (7)
- System Description Languages (7)

Logic (62)

- Philosophy (7)
- Rewriting (11)

Mathematics (199)

- Order (6)
- Algebra (63)
- Analysis (36)
- Probability Theory (12)
- Number Theory (26)
- Economics (10)
- Geometry (17)
- Topology (4)
- Graph Theory (16)
- Combinatorics (18)
- Category Theory (6)
- Physics (1)
- Set Theory (1)
- Misc (3)

Tools (14)

Archive of Formal Proofs III

- New logic entry:
  - Hybrid Logic – Formalizing a Seligman-Style Tableau System
  - Based on work by Blackburn, Bolander, Braüner and Jørgensen.
  - https://www.isa-afp.org/entries/Hybrid_Logic.html
- “[V]ery nice and clean proofs!”
  - Gerwin Klein, AFP editor
- Latest version:
  - 4820 lines of Isar proof code.
  - 100+ pages when rendered in LaTeX.
  - Checked in less than two minutes on my machine.

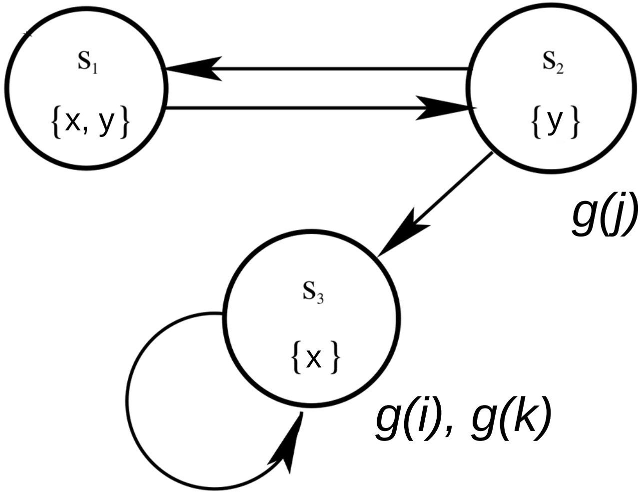
Hybrid Logic

- Modal logic enriched with names for worlds, nominals (a, b, c, i, j, k).
- Nominals give rise to the satisfaction operator (@).
- Semantics defined over an *assignment (g)*, a mapping from nominals to worlds.

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

```
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)›
```



Seligman-Style Tableau System I

- Syntactic procedure for proving validities.
- A tableau closes if we can apply rules to reach a contradiction on all branches.
- Division of each branch into blocks
  - Opening nominal acts as prefix/label.
  - Intuition: Formulas on a block are true in the world denoted by the opening nominal.
  - Can be viewed as a macro.
- Rules operate on arbitrary formulas (within blocks).
- If φ occurs on an a-block, I say that φ occurs *at a*.

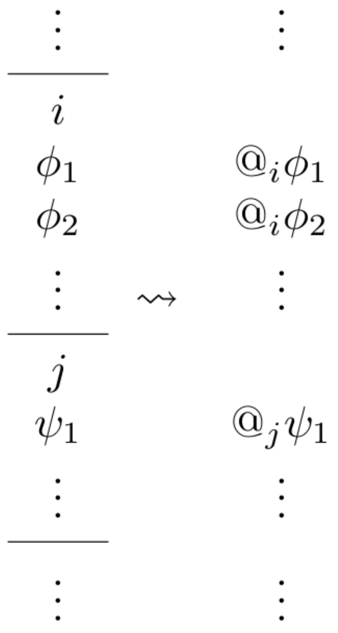


Figure 3.1: Blocks as macros.

Seligman-Style Tableau System II

$$\frac{\begin{array}{c} a \\ \phi \vee \psi \end{array}}{\begin{array}{c} a \\ / \quad \backslash \\ \phi \qquad \psi \end{array}}$$

$(\vee)$

$$\frac{\begin{array}{c} a \\ \neg(\phi \vee \psi) \end{array}}{\begin{array}{c} a \\ | \\ \neg\phi \\ \neg\psi \end{array}}$$

$(\neg\vee)$

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

$(\neg\neg)$

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

$(\Diamond)^1$

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

$(\neg\Diamond)$

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

GoTo[2]

$$\frac{\begin{array}{ccc} b & b & a \\ i & \phi & i \end{array}}{\begin{array}{c} a \\ | \\ \phi \end{array}}$$

Nom

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

Closing

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

$(@)$

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

$(\neg @)$

[1] i is fresh, ϕ is not a nominal.
[2] i is not fresh.

Restrictions Towards Termination

- **R1** The output of a rule must include a formula *new* to the current block type.
  - Reformulated to be explicit about rule applicability, not output.
- **R2** The ($\Diamond$) rule can only be applied to input $\Diamond\varphi$ on an a-block if $\Diamond\varphi$ is not already *witnessed* at a.
  - $\Diamond\varphi$ is *witnessed* at a if for some i both $\Diamond i$ and $@i\ \varphi$ occur at a.
  - Reformulated in terms of branch content, not rule applications.
- ~~**R3** The Name rule is only ever applied as the very first rule in a tableau.~~
- **R4** The GoTo rule consumes one coin from the bank. (Other rules add one coin.)
  - Reformulated to make rule induction easier.
- **R5** (@) and (¬@) can only be applied to premises i and $@i\ \varphi$ ($\neg @i\ \varphi$) when the current block is an i-block.
  - Incorporated structurally. Original versions admissible.

Tricky Example for Original Nom

1.	a	
2.	$\neg @_i(j \to @_j(i \to @_i(\phi \to @_j\phi)))$	
3.	i	GoTo
4.	$\neg(j \to @_j(i \to @_i(\phi \to @_j\phi)))$	$(\neg @)$ 2, 3
5.	j	$(\neg \to)$ 4
6.	$\neg @_j(i \to @_i(\phi \to @_j\phi))$	$(\neg \to)$ 4
7.	j	GoTo
8.	$\neg(i \to @_i(\phi \to @_j\phi))$	$(\neg @)$ 6, 7
9.	i	$(\neg \to)$ 8
10.	$\neg @_i(\phi \to @_j\phi)$	$(\neg \to)$ 8
11.	i	GoTo
12.	$\neg(\phi \to @_j\phi)$	$(\neg @)$ 10, 11
13.	ϕ	$(\neg \to)$ 12
14.	$\neg @_j\phi$	$(\neg \to)$ 12
15.	j	GoTo
16.	$\neg\phi$	$(\neg @)$ 14, 15

Figure 3.6: Getting stuck with R1+R5 and $(\neg \to)$.

1.	a	
2.	$\neg @_i(\neg j \vee @_j(\neg i \vee @_i(\neg\phi \vee @_j\phi)))$	
3.	i	GoTo
4.	$\neg(\neg j \vee @_j(\neg i \vee @_i(\neg\phi \vee @_j\phi)))$	$(\neg @)$ 2, 3
5.	$\neg\neg j$	$(\neg\vee)$ 4
6.	$\neg @_j(\neg i \vee @_i(\neg\phi \vee @_j\phi))$	$(\neg\vee)$ 4
7.	j	GoTo
8.	$\neg(\neg i \vee @_i(\neg\phi \vee @_j\phi))$	$(\neg @)$ 6, 7
9.	$\neg\neg i$	$(\neg\vee)$ 8
10.	$\neg @_i(\neg\phi \vee @_j\phi)$	$(\neg\vee)$ 8
11.	i	GoTo
12.	$\neg(\neg\phi \vee @_j\phi)$	$(\neg @)$ 10, 11
13.	$\neg\neg\phi$	$(\neg\vee)$ 12
14.	$\neg @_j\phi$	$(\neg\vee)$ 12
15.	j	GoTo
16.	$\neg\phi$	$(\neg @)$ 14, 15
17.	i	$(\neg\neg)$ 9
18.	$\neg\neg\phi$	Nom 11, 13, 17
	$\times$	

Figure 3.7: Being saved from R1 by double negations.

Tableau System in Isabelle

```
inductive ST :: ‹nat ⇒ ('a, 'b) branch ⇒ bool› (‹_ ⊢ _› [50, 50] 50) where
  Close:
  ‹p at i in branch ⟹ (¬ p) at i in branch ⟹
   n ⊢ branch›
| Neg:
  ‹(¬ ¬ p) at a in (ps, a) # branch ⟹
   new p a ((ps, a) # branch) ⟹
   Suc n ⊢ (p # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| DisP:
  ‹(p ∨ q) at a in (ps, a) # branch ⟹
   new p a ((ps, a) # branch) ⟹ new q a ((ps, a) # branch) ⟹
   Suc n ⊢ (p # ps, a) # branch ⟹ Suc n ⊢ (q # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| DisN:
  ‹(¬ (p ∨ q)) at a in (ps, a) # branch ⟹
   new (¬ p) a ((ps, a) # branch) ∨ new (¬ q) a ((ps, a) # branch) ⟹
   Suc n ⊢ ((¬ q) # (¬ p) # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| DiaP:
  ‹(◇ p) at a in (ps, a) # branch ⟹
   i ∉ branch_nominals ((ps, a) # branch) ⟹
   ∄a. p = Nom a ⟹ ¬ witnessed p a ((ps, a) # branch) ⟹
   Suc n ⊢ ((@ i p) # (◇ Nom i) # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| DiaN:
  ‹(¬ (◇ p)) at a in (ps, a) # branch ⟹
   (◇ Nom i) at a in (ps, a) # branch ⟹
   new (¬ (@ i p)) a ((ps, a) # branch) ⟹
   Suc n ⊢ ((¬ (@ i p)) # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| SatP:
  ‹(@ a p) at b in (ps, a) # branch ⟹
   new p a ((ps, a) # branch) ⟹
   Suc n ⊢ (p # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| SatN:
  ‹(¬ (@ a p)) at b in (ps, a) # branch ⟹
   new (¬ p) a ((ps, a) # branch) ⟹
   Suc n ⊢ ((¬ p) # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
| GoTo:
  ‹i ∈ branch_nominals branch ⟹
   n ⊢ ([], i) # branch ⟹
   Suc n ⊢ branch›
| Nom:
  ‹p at b in (ps, a) # branch ⟹ Nom i at b in (ps, a) # branch ⟹
   Nom i at a in (ps, a) # branch ⟹
   new p a ((ps, a) # branch) ⟹
   Suc n ⊢ (p # ps, a) # branch ⟹
   n ⊢ (ps, a) # branch›
```

Soundness

```
lemma soundness:
  assumes ‹n ⊢ branch›
  shows ‹∃block ∈ set branch. ∃p on block. ¬ M, g, w ⊨ p›
```

```
theorem soundness_fresh:
  assumes ‹n ⊢ [([¬ p], i)]› ‹i ∉ nominals p›
  shows ‹M, g, w ⊨ p›
proof -
  from assms(1) have ‹M, g, g i ⊨ p› for g
    using soundness by fastforce
  then have ‹M, g(i := w), (g(i := w)) i ⊨ p›
    by blast
  then have ‹M, g(i := w), w ⊨ p›
    by simp
  then have ‹M, g(i := g i), w ⊨ p›
    using assms(2) semantics_fresh by metis
  then show ?thesis
    by simp
qed
```

No Detours I

- Originally [**R4** The GoTo rule cannot be applied twice in a row.]

- On the weakened branch, opening the *a*-block was a mistake.
- Pruning detours complicates proofs.
- Instead, assume more coins.
- Show separately that a single initial coin is sufficient.
- Coin system allows for detours but ties GoTo to initial savings or other rule applications, i.e. progress.

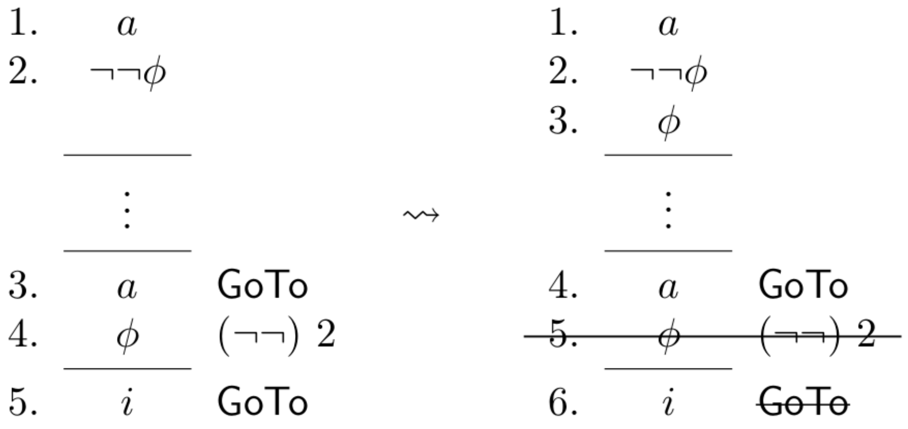


Figure 3.9: Unjustified GoTo after weakening.

No Detours II

LEMMA 4.3 (FILTERING DETOURS) *If a branch can be closed starting from n coins, then any filtering cut of the branch can be closed from $m+1$ coins. That is, if $n \vdash \Theta_l, \Theta_r$ then $m+1 \vdash \lfloor \Theta_l \rfloor, \Theta_r$.*

THEOREM 4.4 (POSITIVE COINS) *If $n \vdash \Theta$ then $m+1 \vdash \Theta$.*

COROLLARY 4.5 (A SINGLE COIN) *If $n \vdash \Theta$ then $1 \vdash \Theta$.*

THEOREM 4.6 (FREE GOTO)
If $n+1 \vdash B, \Theta$ where B is an empty block whose opening nominal occurs in Θ, then $n+1 \vdash \Theta$.

PROOF. By applying GoTo we have $n+2 \vdash \Theta$ and then Theorem 4.4 gives us $n+1 \vdash \Theta$ as wanted. □

As Good as New

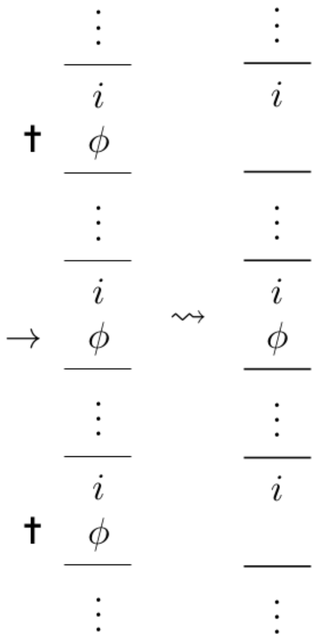


Figure 4.1: Strengthening.

- Mark one lasting occurrence of *φ at i* and any number of removed occurrences.
- When a removed occurrence is used as rule input, the lasting one can be used instead.
- Allows us to lift **R1**.
- Requires indexing machinery.
- Alternatively: Cut branch and remove every occurrence of *φ at i* below the cut. Would require cutting in the middle of block.

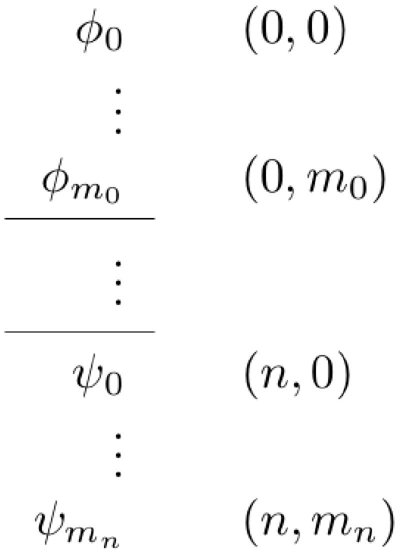


Figure 4.2: Indexing.

Too Many Witnesses I

THEOREM 4.10 (SUBSTITUTION) *Let θ be a substitution function. Assume that for all finite sets A, if there exists a nominal not in A then there exists a nominal not in the image of A under f. If $\vdash \Theta$ then $\vdash \Theta\theta$.*

PROOF. Shown by rule induction over the construction of Θ for an arbitrary θ.

Case ($\Diamond$) By assumption we have $\Diamond\phi$ at a in Θ, the nominal i is fresh in Θ and by the induction hypothesis $\vdash (@_i\phi)\theta' -_{\theta'(a)} (\Diamond i)\theta' -_{\theta'(a)} \Theta\theta'$ for any θ'. The $\Diamond\phi$ is unwitnessed at a in Θ but since the substitution may collapse formulas, $\Diamond\phi\theta$ may be witnessed at $\theta(a)$ in $\Theta\theta$. Thus there are two cases:

If $\Diamond\phi\theta$ is witnessed at $\theta(a)$ in $\Theta\theta$ then let i' be the witnessing nominal, such that $@_{i'}(\phi\theta)$ and $\Diamond i'$ both occur at $\theta(a)$ in $\Theta\theta$. Apply the induction hypothesis at $\theta(i := i')$ to obtain $\vdash @_{i'}(\phi\theta) -_{\theta(a)} \Diamond i' -_{\theta(a)} \Theta\theta$, where the added assignment has been reduced away in the places where i is fresh. Both formulas in the extension are justified by the Nom rule so we obtain $\vdash \Theta\theta$ as needed.

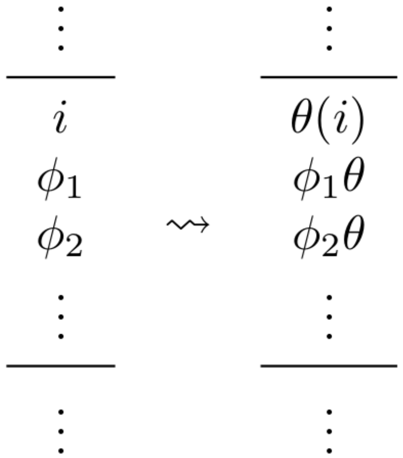


Figure 4.3: Substitution.

Too Many Witnesses II

Otherwise the formula is unwitnessed. To apply the ($\Diamond$) rule, we need the witnessing nominal to be fresh in $\Theta\theta$ but since θ is not necessarily injective, this may not be the case for $\theta(i)$. But since Θ is finite, we have by assumption a nominal j that is fresh to $\Theta\theta$. Apply the induction hypothesis at $\theta(i := j)$ to learn $\vdash @_j(\phi\theta) -_{\theta(a)} \Diamond j -_{\theta(a)} \Theta\theta$ where, again, I have reduced the term using the fact that i is fresh in Θ. The ($\Diamond$) rule now applies: $\Diamond\phi\theta$ is unwitnessed at $\theta(a)$ in $\Theta\theta$ and we have ensured that j is fresh. Thus we can conclude $\vdash \Theta\theta$. $\Box$

Theorem 4.13 (Unrestricted ($\Diamond$)) *If $\vdash @_i\phi -_a \Diamond i -_a \Theta$, i is fresh in Θ and ϕ is not a nominal, then $\vdash \Theta$.*

General Satisfaction I

$$\begin{array}{c} B_1 \\ \hline B_2 \\ \hline \vdots \\ \hline B_n \end{array} \quad \rightsquigarrow \quad \begin{array}{c} B'_1 \\ \hline B'_2 \\ \hline \vdots \\ \hline B'_m \end{array}$$

Figure 4.4: Rearranging.

$$\{B_1, B_2, \ldots, B_n\} \subseteq \{B'_1, B'_2, \ldots, B'_m\}$$

- Rearranged branch may have mismatched current block.
- Apply induction hypothesis at the branch extended by the original current block.
- Then drop it again.
- Custom induction principle for dropping.
- Need substitution for (◇) case.

```
lemma list_down_induct [consumes 1, case_names Start Cons]:
  assumes ‹∀y ∈ set ys. Q y› ‹P (ys @ xs)›
    ‹⋀y xs. Q y ⟹ P (y # xs) ⟹ P xs›
  shows ‹P xs›
  using assms by (induct ys) auto
```

	$\vdots$	
1.	a	
	$\vdots$	
2.	ϕ_1	
	$\vdots$	
3.	ϕ_2	
	$\vdots$	
4.	a	GoTo
5.	ϕ_1	Nom 1, 2, 4
6.	ϕ_2	Nom 1, 3, 4
	$\vdots$	$\vdots$

Figure 4.5: Dropping a block.

General Satisfaction II

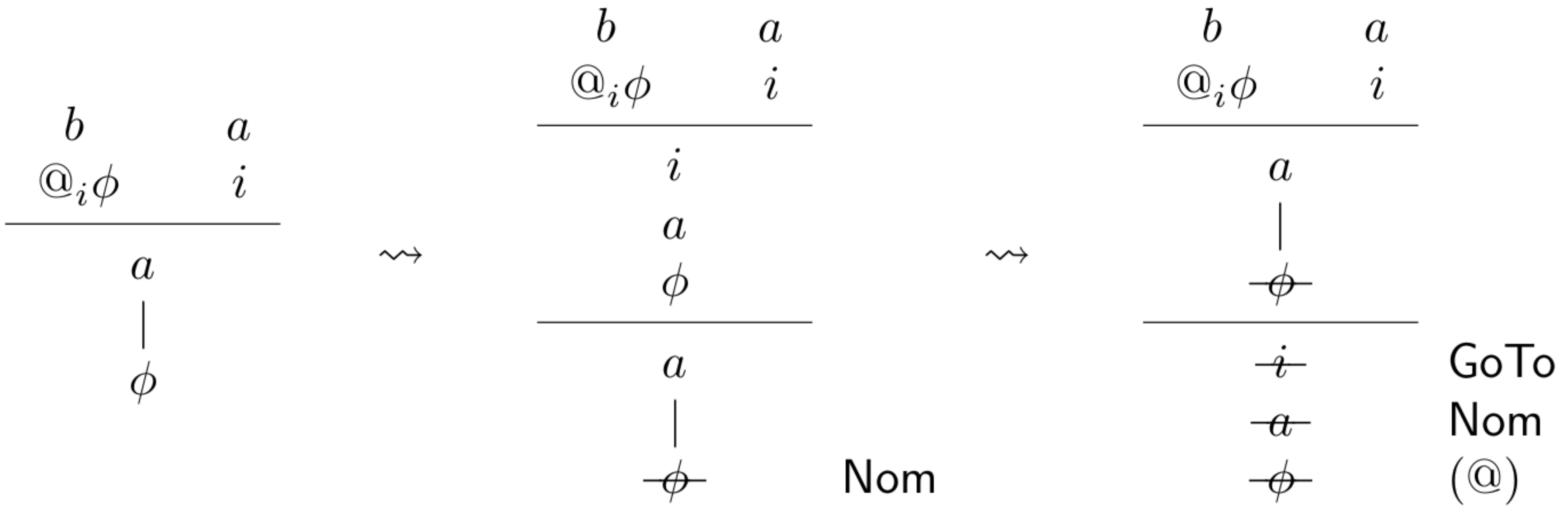
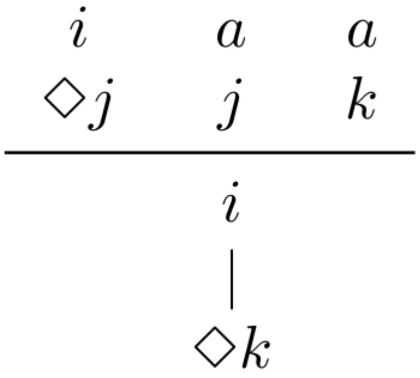


Figure 4.6: Deriving the unrestricted (@) rule.

Bridge I

- Replace $\Diamond k$ by $\Diamond j$ and apply Nom.
- Need to update any output of rules with $\Diamond k$ as input.
- Lemma 4.2 by Jørgensen et al. but with a descendant set.
  - Defined by branch content, not rule applications.
  - Lemma 4.1 (shape of descendants) comes for free.



Definition 5.1 (Descendant set)

Initial *If* $\Theta(v)$ *is an* i*-block and* $\Theta(v, v') = \Diamond k$*, then* $\{(v, v')\}$ *is* $D_{\Theta,i,k}$.

Derived *If* Δ *is* $D_{\Theta,i,k}$*,* $(w, w') \in \Delta$*,* $\Theta(w)$ *is an* a*-block,* $\Theta(w, w') = \Diamond k$*,* $\Theta(v)$ *is an* a*-block and* $\Theta(v, v') = \neg @_k \phi$ *for some* ϕ *then* $\{(v, v')\} \cup \Delta$ *is* $D_{\Theta,i,k}$.

Copied *If* Δ *is* $D_{\Theta,i,k}$*,* $(w, w') \in \Delta$*,* $\Theta(w)$ *is a* b*-block,* $\Theta(w, w') = \phi$*, there is a nominal* j *that occurs both at* a *and* b *in* Θ*,* $\Theta(v)$ *is an* a*-block and* $\Theta(v, v') = \phi$ *then* $\{(v, v')\} \cup \Delta$ *is* $D_{\Theta,i,k}$.

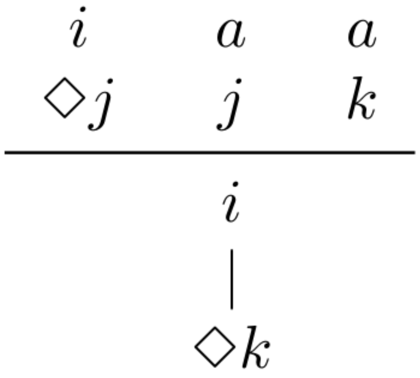
Bridge II

Case $(\neg\Diamond)$ We have both $\neg\Diamond\phi$ and $\Diamond i'$ at a in Θ, the current block is an a-block and we know that Δ is $D_{\Theta,i,k}$. Let (w, w') be the index of the given $\Diamond i'$. There are two cases.

If $(w, w') \notin \Delta$ then $\Diamond i'$ is also at a in Θ^{Δ} and the case follows similarly to $(\neg\neg)$: The rule input is unchanged so we do not have to replace the output.

Otherwise, $(w, w') \in \Delta$ so by Lemma 5.2 on page 40, $i' = k$ and $(\Diamond i')^j = \Diamond j$. To account for this, we need to extend Δ to include the index of the output, $\neg @_k \phi$, to make sure it becomes $\neg @_j \phi$ such that the $(\neg\Diamond)$ rule justifies it. Let (v, v') be the index of the output. By Lemma 5.3 on the previous page, Δ is $D_{\neg @_k \phi -_a \Theta, i, k}$ and by the **Derived** case, so is $\{(v, v')\} \cup \Delta$.

Thus we apply the induction hypothesis at the extended index set, $\{(v, v')\} \cup \Delta$, and learn $\vdash (\neg @_k \phi)^j -_a \Theta^{j_{\{(v,v')\}\cup\Delta}}$. By Remark 5.4 on the preceding page we have $\vdash \neg @_j \phi -_a \Theta^{\Delta}$. We can now apply the $(\neg\Diamond)$ rule and conclude the case.



Bridge III

```
theorem Bridge:
  fixes i :: 'b
  assumes inf: ‹infinite (UNIV :: 'b set)› and
    ‹Nom i at b in branch› ‹(◇ Nom j) at b in branch› ‹Nom i at a in branch›
    ‹Nom j at c in branch› ‹Nom k at c in branch›
    ‹⊢ ((◇ Nom k) # ps, a) # branch›
  shows ‹⊢ (ps, a) # branch›
proof -
  let ?xs = ‹{(length branch, length ps)}›

  have ‹descendants k a (((◇ Nom k) # ps, a) # branch) ?xs›
    using Initial by force
  moreover have
    ‹Nom j at c in ((◇ Nom k) # ps, a) # branch›
    ‹Nom k at c in ((◇ Nom k) # ps, a) # branch›
    using assms(5-6) by auto
  ultimately have
    ‹⊢ mapi_branch (bridge k j ?xs) (((◇ Nom k) # ps, a) # branch)›
    using ST_bridge inf assms(7) by fast
  then have ‹⊢ ((◇ Nom j) # mapi (bridge k j ?xs (length branch)) ps, a) #
      mapi_branch (bridge k j ?xs) branch›
    unfolding mapi_branch_def by simp
  moreover have ‹mapi_branch (bridge k j {(length branch, length ps)}) branch =
      mapi_branch (bridge k j {}) branch›
    using mapi_branch_add_oob[where xs=‹{}›] by fastforce
  moreover have ‹mapi (bridge k j ?xs (length branch)) ps =
    mapi (bridge k j {} (length branch)) ps›
    using mapi_block_add_oob[where xs=‹{}› and ps=ps] by simp
  ultimately have ‹⊢ ((◇ Nom j) # ps, a) # branch›
    using mapi_block_id[where ps=ps] mapi_branch_id[where branch=branch] by simp
  then show ?thesis
    using assms(2-5) by (meson Nom' set_subset_Cons subsetD)
qed
```

THEOREM 5.7
If $\Diamond j$ is at i in Θ, j and k are both at a in Θ, the current block is an i-block and $\vdash \Diamond k -_i \Theta$, then $\vdash \Theta$.

PROOF. Let (v, v') be the index of the final $\Diamond k$ and let Δ be the set containing just that index. By the **Initial** case, Δ is $D_{\Diamond k -_i \Theta, i, k}$. Thus $\vdash (\Diamond k)^j -_i \Theta^\Delta$ by Lemma 5.6 on page 41. Then $\vdash \Diamond j -_i \Theta$ by Lemma 5.4 on page 41 and finally, due to the Nom rule, $\vdash \Theta$. □

Completeness

```
lemma hintikka_model:
  assumes ‹hintikka H›
  shows
    ‹p at i in' H ⟹ Model (reach H) (val H), assign H, assign H i ⊨ p›
    ‹(¬ p) at i in' H ⟹ ¬ Model (reach H) (val H), assign H, assign H i ⊨ p›

primrec extend ::
  ‹('a, 'b) block set ⇒ (nat ⇒ ('a, 'b) block) ⇒ nat ⇒ ('a, 'b) block set› where
  ‹extend S f 0 = S›
| ‹extend S f (Suc n) =
    (if ¬ consistent ({f n} ∪ extend S f n)
     then extend S f n
     else
      let used = (⋃block ∈ {f n} ∪ extend S f n. block_nominals block)
      in {f n, witness (f n) used} ∪ extend S f n)›
```

```
lemma hintikka_Extend:
  fixes S :: ‹('a, 'b) block set›
  assumes inf: ‹infinite (UNIV :: 'b set)› and
    ‹maximal S› ‹consistent S› ‹saturated S›
  shows ‹hintikka S›
```

```
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 ‹1 ⊢ [([¬ p], i)]›
```

Hintikka

- Small error in the Hintikka definition by Jørgensen et al.
- Worlds of the named model are sets of equivalent nominals, $|i|$.
- But their base case does not account for this.

(i) *If there is an* i*-block in* H *with an atomic formula* a *on it, then there is no* i*-block in* H *with* $\neg a$ *on it.*

- The world $|i|$ ($|j|$) supposedly models both x and its negation:

$$\{([i, x], j), ([j, \neg x], i)\}$$

(i') If there is an i-block in H with j on it and a j-block in H with a propositional symbol x on it, then there is no i-block in H with $\neg x$ on it.

Hintikka Definition

definition *hintikka* :: ‹('a, 'b) block set ⇒ bool› **where**

$$
\begin{aligned}
&\langle hintikka\ H \equiv \\
&\quad (\forall x\ i\ j.\ Nom\ j\ at\ i\ in'\ H \longrightarrow Pro\ x\ at\ j\ in'\ H \longrightarrow \\
&\qquad \neg\ (\neg\ Pro\ x)\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall a\ i.\ Nom\ a\ at\ i\ in'\ H \longrightarrow \neg\ (\neg\ Nom\ a)\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall i\ j.\ (\Diamond\ Nom\ j)\ at\ i\ in'\ H \longrightarrow \neg\ (\neg\ (\Diamond\ Nom\ j))\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ i.\ i \in nominals\ p \longrightarrow (\exists\, block \in H.\ p\ on\ block) \longrightarrow \\
&\qquad (\exists\, ps.\ (ps,\ i) \in H)) \\
&\wedge \\
&\quad (\forall i\ j.\ Nom\ j\ at\ i\ in'\ H \longrightarrow Nom\ i\ at\ j\ in'\ H) \\
&\wedge \\
&\quad (\forall i\ j\ k.\ Nom\ j\ at\ i\ in'\ H \longrightarrow Nom\ k\ at\ j\ in'\ H \longrightarrow \\
&\qquad Nom\ k\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall i\ j\ k.\ (\Diamond\ Nom\ j)\ at\ i\ in'\ H \longrightarrow Nom\ k\ at\ j\ in'\ H \longrightarrow \\
&\qquad (\Diamond\ Nom\ k)\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall i\ j\ k.\ (\Diamond\ Nom\ j)\ at\ i\ in'\ H \longrightarrow Nom\ k\ at\ i\ in'\ H \longrightarrow \\
&\qquad (\Diamond\ Nom\ j)\ at\ k\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ q\ i.\ (p \vee q)\ at\ i\ in'\ H \longrightarrow \\
&\qquad p\ at\ i\ in'\ H \vee q\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ q\ i.\ (\neg\ (p \vee q))\ at\ i\ in'\ H \longrightarrow \\
&\qquad (\neg\ p)\ at\ i\ in'\ H \wedge (\neg\ q)\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ i.\ (\neg\ \neg\ p)\ at\ i\ in'\ H \longrightarrow \\
&\qquad p\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ i\ a.\ (@\ i\ p)\ at\ a\ in'\ H \longrightarrow \\
&\qquad p\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ i\ a.\ (\neg\ (@\ i\ p))\ at\ a\ in'\ H \longrightarrow \\
&\qquad (\neg\ p)\ at\ i\ in'\ H) \\
&\wedge \\
&\quad (\forall p\ i.\ (\nexists\, a.\ p = Nom\ a) \longrightarrow (\Diamond\ p)\ at\ i\ in'\ H \longrightarrow \\
&\qquad (\exists j.\ (\Diamond\ Nom\ j)\ at\ i\ in'\ H \wedge (@\ j\ p)\ at\ i\ in'\ H)) \\
&\wedge \\
&\quad (\forall p\ i\ j.\ (\neg\ (\Diamond\ p))\ at\ i\ in'\ H \longrightarrow (\Diamond\ Nom\ j)\ at\ i\ in'\ H \longrightarrow \\
&\qquad (\neg\ (@\ j\ p))\ at\ i\ in'\ H)\rangle
\end{aligned}
$$

Future Work

- Restricting Nom for termination
  - The thesis sketches an idea based on tags that encode the notion of one nominal being *generated* by another, forcing a direction on Nom.
- Proving and formalizing termination directly by a decreasing length argument instead of by translation.
- Verifying an algorithm, a decision procedure, based on the calculus.
  - And using Isabelle to generate executable code based on it.
- Extending the formalization to prove more results about hybrid logic, e.g. interpolation
- Giving an internalized restriction on GoTo instead of the coin system.

Conclusion

- I have formalized the soundness and completeness of a tableau system for basic hybrid logic in Isabelle/HOL.
- I have reformulated existing termination restrictions to ease formalization.
- I have shown how to lift the restrictions by working within the system.
  - This simplifies the application of an existing synthetic completeness proof.
- I have shown that the full Bridge rule is admissible.
- The work has been accepted into the Archive of Formal Proofs.

References

- Patrick Blackburn, Thomas Bolander, Torben Braüner, and Klaus Frovin Jørgensen. Completeness and Termination for a Seligman-style Tableau System. *Journal of Logic and Computation*, 27(1):81–107, 2017.
- Klaus Frovin Jørgensen, Patrick Blackburn, Thomas Bolander, and Torben Braüner. Synthetic Completeness Proofs for Seligman-style Tableau Systems. In *Advances in Modal Logic 11*, pages 302–321, 2016.
- Torben Braüner. Hybrid Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition, 2017.
- Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

See the thesis for the rest.