Belief Revision and Isabelle/HOL

Asta Halkjær From, DTU Compute

2020-11-26
(revised later)

What we talk about when we talk about a logic
or
(the unexpected virtue of lambda calculus)

Agenda

- **Belief revision**
- Isabelle
- Isabelle/HOL
- Case study
- Demo

Background

Belief states: *s*

Possible worlds: *x, y, z*

Propositions: *A, B, C,* $\top$, $\bot$

Predicates on worlds (*A x*)

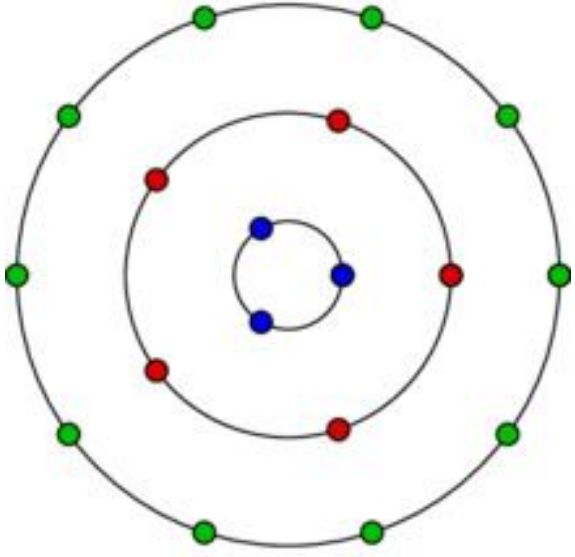
Consistent if satisfied by some world

Belief ordering functions (bof): $\prec$

$x \prec_s y$, *x* is more plausible than *y* in s

Belief change functions (bcf): $*$, $\div$

revision: s $*$ A, B, contraction: s $\div$ A, B



Varieties of Orderings

Strict Partial Order (SPO):

- Irreflexive: $\neg(x \prec_s x)$
- Transitive: if $x \prec_s y$ and $y \prec_s z$ then $x \prec_s z$

Interval Order (IO):

- SPO
- Inter: if $x \prec_s y$ and $z \prec_s w$ then $x \prec_s w$ or $z \prec_s y$

Semiorder (SO):

- IO
- Semi: …

Modular Order (MO): ...

Varieties of Revision and Contraction

World x is minimal wrt. bof $\lessdot_s$ and proposition A: $\min(\lessdot_s, A)(x)$
for all worlds y: if $A\ y$ then $\neg\ (y \lessdot_s x)$

Minimal revision: MinR($*$, $\lessdot$)
for all x, s, consistent A: $\min(\lessdot_{s*\top, A}, \top)(x)$ iff $\min(\lessdot_s, A)(x)$

Minimal contraction: MinC($\div$, $\lessdot$) …

Order-reductive revision: ORedR($*$, $\lessdot$) …

DP revision …

…

Sample Theorems

Theorem 1. $\forall \prec . (MO(\prec) \rightarrow (SO(\prec) \wedge IO(\prec)))$

Theorem 2. $\exists \prec . (SO(\prec) \wedge \neg IO(\prec))$

Theorem 3. $\exists \prec . (IO(\prec) \wedge \neg SO(\prec))$

Theorem 4. $\forall \prec, *. ((MO(\prec) \wedge MinR(*, \prec) \wedge (NatR(*, \prec) \vee ResR(*, \prec) \vee LexR(*, \prec))) \rightarrow DPR(*, \prec))$

Theorem 5. $\forall \prec, *. ((MO(\prec) \wedge MinR(*, \prec) \wedge ((ResR(*, \prec) \vee LexR(*, \prec))) \rightarrow AdmR(*, \prec))$

Theorem 6. $\forall \prec, *. ((MO(\prec) \wedge MinR(*, \prec) \wedge POIR(*, \prec)) \rightarrow IIAI(*, \prec))$

Theorem 7. $\forall \prec, *. ((MO(\prec) \wedge MinR(*, \prec)) \rightarrow (ElemR(*, \prec) \leftrightarrow (NatR(*, \prec) \vee ResR(*, \prec) \vee LexR(*, \prec)))$

Theorem 8. $\forall \prec, *. ((MO(\prec) \wedge MinR(*, \prec) \wedge IIAPR(*, \prec)) \rightarrow ORedR(*, \prec))$

Computer, Please Help Me

> *My colleagues and I working in the field of belief change make a large number of conjectures about the satisfiability of various properties by tuples of binary relations (e.g. total preorders, interval orders, semiorders, etc.) over finite sets of objects. The reason for this [project], of course, is that* ***we want to check for countermodels*** *before going through the hassle of attempting to prove a certain conjecture.*
>
> *The formulae whose models we are looking for are* ***most conveniently expressed in higher-order logic****: they involve quantification over higher order-relations. Since we are working with finite models, the problem can be reduced to propositional logic. But this is a pain.*

Jake Chandler, Automated Reasoning for Belief Revision (Notes)

Agenda

- Belief revision
- **Isabelle**
- Isabelle/HOL
- Case study
- Demo

Generic Proof Assistant

Isabelle is a framework

- Provides the glue
- You can pick the logic

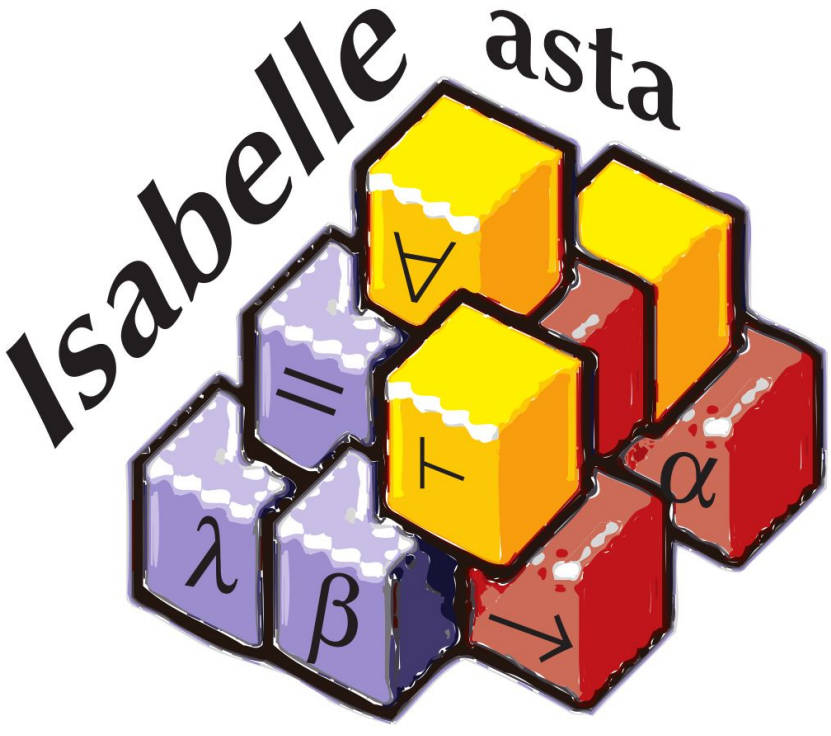
Glue?

$\forall x.\ P\ x \rightarrow P\ x$

$\forall x.\ \rightarrow (P\ x)\ (P\ x)$

$\forall (\lambda x.\ \rightarrow (P\ x)\ (P\ x))$

$\wedge(\lambda P.\ \forall(\lambda x.\ \rightarrow (P\ x)\ (P\ x)))$



Generic Proof Assistant

Isabelle is a framework

- Provides the glue
- You can pick the logic

Glue?

$\forall x.\ P\ x \rightarrow P\ x$

$\forall x.\ \rightarrow (P\ x)\ (P\ x)$

$\forall (\lambda x.\ \rightarrow (P\ x)\ (P\ x))$

$\wedge(\lambda P.\ \forall(\lambda x.\ \rightarrow (P\ x)\ (P\ x)))$

Typed lambda calculus:

- variable: x
- abstraction: λx. M
- application: M N
- \+ constants

Example types:

- x : i
- P x : o

Example constants:

- $\rightarrow : o \Rightarrow o \Rightarrow o$
- $\forall : (i \Rightarrow o) \Rightarrow o$

Isabelle/Pure I

The basic building blocks

Typed lambda calculus:

- Single base type “prop”
- Function type constructor ⇒

Constants:

- ⋀ : (‘a ⇒ prop) ⇒ prop
- ⟹ : prop ⇒ prop ⇒ prop
- ≡ : ‘a ⇒ ‘a ⇒ prop

\+ introduction and elimination rules that give *meaning* (see src/Pure/thm.ml)

Isabelle/Pure II

[Axioms for ≡]

Functions:

$$\frac{\begin{array}{c}[x :: \alpha] \\ \vdots \\ b(x) :: \beta\end{array}}{\lambda x.\ b(x) :: \alpha \Rightarrow \beta}\ (\Rightarrow I) \qquad \frac{b :: \alpha \Rightarrow \beta \quad a :: \alpha}{b\ a :: \beta}\ (\Rightarrow E)$$

Meta-quantifier:

$$\frac{\begin{array}{c}[x] \\ \vdots \\ B(x)\end{array}}{\bigwedge x.\ B(x)}\ (\bigwedge I) \qquad \frac{\bigwedge x.\ B\ x}{B\ a}\ (\bigwedge E)$$

Meta-implication:

$$\frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Longrightarrow B}\ (\Longrightarrow I) \qquad \frac{A \Longrightarrow B \quad A}{B}\ (\Longrightarrow E)$$

Isabelle/IFOL I

Our own propositions:

```
typedecl o

judgment Trueprop :: "o ⇒ prop"  ("_" 5)
```

Conjunction explained in terms of Pure building blocks:

```
axiomatization conj :: "o ⇒ o ⇒ o"  (infixr "∧" 35)
   where conjI [intro]: "A ⟹ B ⟹ A ∧ B"
      and conjD1: "A ∧ B ⟹ A"
      and conjD2: "A ∧ B ⟹ B"
```

+ other connectives

$$\frac{\phi \qquad \psi}{\phi \wedge \psi} \wedge I$$

$$\frac{\phi \wedge \psi}{\phi} \wedge E_1$$

$$\frac{\phi \wedge \psi}{\psi} \wedge E_2$$

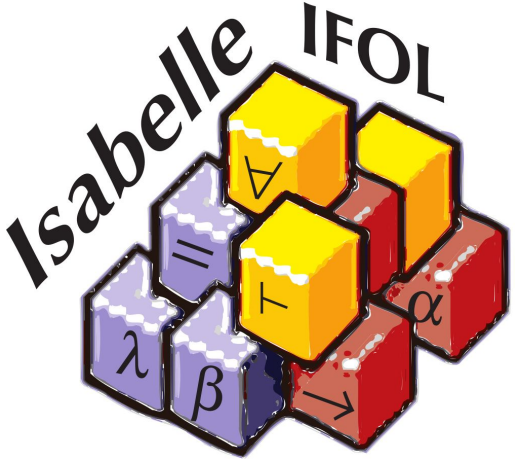
Isabelle/IFOL II

First-order logic has a domain of discourse (individuals):

```
typedecl i
```

Quantification is over that domain:

```
axiomatization All :: "(i ⇒ o) ⇒ o"  (binder "∀" 10)
   where allI [intro]: "(⋀x. P x) ⟹ ∀x. P x"
      and allD [dest]: "∀x. P x ⟹ P a"
```



Isabelle now “speaks” intuitionistic first-order logic

+ precision (not just a natural language explanation)

+ computer assistance

Isabelle/Isar

Rule applications hidden behind *Intelligible semi-automated reasoning* syntax

First **proof** runs *impI*:

- ⟶ becomes ⟹
- **assume** is ⟹I
- goal changes to rhs

Same syntax for any logic

“human-readable”

impI ***[intro]:*** *"(A ⟹ B) ⟹ A ⟶ B"*

```
lemma "(∃x. ∀y. R x y) ⟶ (∀y. ∃x. R x y)"
proof
  assume "∃x. ∀y. R x y"
  then obtain x where "∀y. R x y" ..
  show "∀y. ∃x. R x y"
  proof
    fix y
    from ‹∀y. R x y› have "R x y" ..
    then show "∃x. R x y" ..
  qed
qed
```

Takeaway

Formalize the formal:

- Alternative to English
- Unambiguous
- Computer assistance

Logic:

- Constants
- w/ meaning
- Isabelle/Pure glue

Lambda calculus!

Agenda

- Belief revision
- Isabelle
- **Isabelle/HOL**
- Case study
- Demo

Higher-Order Logic

*The formulae whose models we are looking for are **most conveniently expressed in higher-order logic**: they involve quantification over higher order-relations.*

What can we quantify over?		What does the quantifier bind?
- First-order logic:		
- individuals	i	∀x. P x
- Second-order logic:		
- individuals	i	∀x. P x
- predicates	i ⇒ o	∀P. P a
- functions	i ⇒ i	∀f. P (f a)
- Higher-order logic		
- anything	‘a	∀＜. ....

Isabelle/HOL I

text ‹
The following theory development illustrates the foundations of Higher-Order Logic. The ``HOL'' logic that is given here resembles @{cite "Gordon:1985:HOL"} and its predecessor @{cite "church40"}, but the order of axiomatizations and defined connectives has been adapted to modern presentations of ‹λ›-calculus and Constructive Type Theory. Thus it fits nicely to the underlying Natural Deduction framework of Isabelle/Pure and Isabelle/Isar.
›

HOL/Isar_Examples/Higher_Order_Logic.thy
by Makarius

Isabelle/HOL II

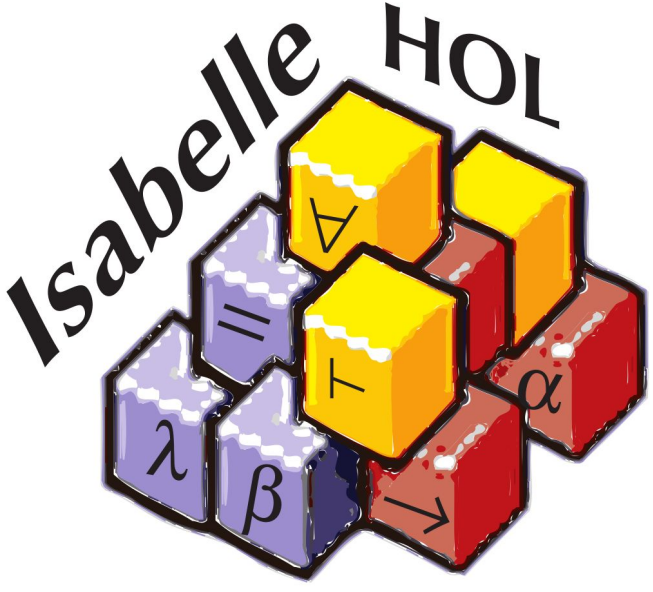
Minimal logic

```
axiomatization imp :: "o ⇒ o ⇒ o"  (infixr "⟶" 25)
   where impI [intro]: "(A ⟹ B) ⟹ A ⟶ B"
     and impE [dest, trans]: "A ⟶ B ⟹ A ⟹ B"

axiomatization All :: "('a ⇒ o) ⇒ o"  (binder "∀" 10)
   where allI [intro]: "(⋀x. P x) ⟹ ∀x. P x"
     and allE [dest]: "∀x. P x ⟹ P a"
```

Define the remaining connectives, e.g.:

```
definition False :: o
   where "False ≡ ∀A. A"
```



Isabelle/HOL III

Prove usual intro/elim rules

Consistency for free

+ Axiomatizations of:

- Equality (=)
- Biconditional (⟷)
- Hilbert's epsilon operator (SOME x. P x)
  (axiom of choice, A ∨ ¬ A)

```
definition conj :: "o ⇒ o ⇒ o"  (infixr "∧" 35)
   where "A ∧ B ≡ ∀C. (A ⟶ B ⟶ C) ⟶ C"

lemma conjI [intro]:
   assumes A and B
   shows "A ∧ B"
   unfolding conj_def
proof
   fix C
   show "(A ⟶ B ⟶ C) ⟶ C"
   proof
      assume "A ⟶ B ⟶ C"
      also note ‹A›
      also note ‹B›
      finally show C .
   qed
qed
```

Takeaway

Foundations:

- expressive language
- simple proof system

Built on top:

- datatype package
- automated provers
- *counterexample search*
- …

Everything boils down to simple, verifiable building blocks

Agenda

- Belief revision
- Isabelle
- Isabelle/HOL
- **Case study**
- Demo

Background Revisited

```
type_synonym ('s, 'w) bof = ‹'s ⇒ 'w ⇒ 'w ⇒ bool›

definition Irref :: ‹('s, 'w) bof ⇒ bool› where
  ‹Irref lt ≡ ∀x s. ¬ lt s x x›

type_synonym 'w prp = ‹'w ⇒ bool›

definition Top :: ‹'w prp› (‹⊤›) where
  ‹Top x ≡ True›

definition min_prp :: ‹('s, 'w) bof ⇒ 's ⇒ 'w prp ⇒ 'w prp› where
  ‹min_prp lt s A x ≡ A x ∧ (∀y. A y ⟶ ¬ lt s y x)›

type_synonym ('s, 'w) bcf = ‹'s ⇒ 'w prp ⇒ 'w prp ⇒ 's›
```

Nitpick I

> *Given a conjecture,* ***Nitpick (via Kodkod and the SAT solver) searches for a standard set-theoretic model that falsifies it*** *while satisfying any relevant axioms and definitions. Nitpick is innately better suited to problems from set theory and logic than Quickcheck. Nitpick revels in particular in finite combinatorial problems.*

Computer, please help me:

```
theorem 1: ‹MO lt ⟶ SO lt ⋀ IO lt›
  nitpick (* "Nitpick found no counterexample" *)
  oops
```

Not a proof, but an indication that the conjecture holds

We can also do a proof

Nitpick II

> *It works by **translating higher-order formulas to first-order relational logic (FORL)** and invoking the highly-optimized SAT-based Kodkod model finder to solve these.*

Search for an example:

```
theorem 3: ‹IO It ∧ ¬ Semi It›
  nitpick[falsify=false]
  (* "Nitpick found a model for card 'a = 1 and card 'b = 4"
     (i.e. such an It exists when there is 1 state and 4 worlds). *)
```

Lots more definitions to formalize

Syntax is suboptimal

Agenda

- Belief revision
- Isabelle
- Isabelle/HOL
- Case study
- **Demo**

Fin

We can be more formal than natural language

This opens the way for computer assistance

We can view logics as constants glued together by lambda calculus

+ meaning via introduction and elimination rules

Higher-order logic is a very simple, powerful idea

References

Paulson, L. C., Nipkow, T., & Wenzel, M. (2019). **From LCF to Isabelle/HOL.** *Formal Aspects of Computing*, 31(6), 675-698.

Blanchette, J. C., & Nipkow, T. (2010, July). **Nitpick: A counterexample generator for higher-order logic based on a relational model finder.** In *International conference on interactive theorem proving* (pp. 131-146). Springer.

https://www.lri.fr/~wolff/tutorials/2014-LRI-isabelle-tutorial/pure.pdf

~~/src/HOL/Isar_Examples/First_Order_Logic.thy

~~/src/HOL/Isar_Examples/Higher_Order_Logic.thy