Can States Be Decidable in Inquisitive Mechanizations?

Implementation of (Bounded) Inquisitive First-Order Logic in Rocq

Max Ole Elliger[1] **Tadeusz Litak**[2]

[1]Friedrich-Alexander-Universität Erlangen-Nürnberg

[2]Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, University of Naples Federico II

September 27, 2025

Intuition

Inquisitive FOL can be seen as an extension of classical logic by questions.

Intuition

Inquisitive FOL can be seen as an extension of classical logic by questions.

Example

Natural Language	Formula
Luisa is guilty.	$\mathrm{Guilty}\,(\mathrm{Luisa})$
If Luisa was there, do we know whether Luisa is guilty?	$\mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$
If we knew whether Luisa was there, do we know whether Luisa is guilty?	$?\,\mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$
Is there some person, who is guilty?	$\exists\, x.\ \mathrm{Guilty}\,(x)$

Intuition

Formulae shall be supported by sets of possible worlds which refer to FO-Models.

Intuition

Formulae shall be supported by sets of possible worlds which refer to FO-Models.

Example

- Consider the following possible worlds regarding Luisa:

	Guilty	**Not Guilty**
Was There	w_1	w_2
Was Not There	w_3	w_4

Intuition

Formulae shall be supported by sets of possible worlds which refer to FO-Models.

Example

- Consider the following possible worlds regarding Luisa:

	Guilty	Not Guilty
Was There	w_1	w_2
Was Not There	w_3	w_4

- We get the following properties regarding the single worlds:

$$w_1 \models \mathrm{WasThere}\,(\mathrm{Luisa}) \rightarrow ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$$
$$w_2 \models \mathrm{WasThere}\,(\mathrm{Luisa}) \rightarrow ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$$
$$w_3 \models \mathrm{WasThere}\,(\mathrm{Luisa}) \rightarrow ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$$
$$w_4 \models \mathrm{WasThere}\,(\mathrm{Luisa}) \rightarrow ?\,\mathrm{Guilty}\,(\mathrm{Luisa})$$

Intuition

Formulae shall be supported by sets of possible worlds which refer to FO-Models.

Example

- Consider the following possible worlds regarding Luisa:

	Guilty	**Not Guilty**
Was There	w_1	w_2
Was Not There	w_3	w_4

- We get the following properties regarding the single worlds:

$$\begin{aligned} w_1 &\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \\ w_2 &\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \\ w_3 &\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \\ w_4 &\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \end{aligned}$$

- If we look at information states, we get the following support properties:

$$\begin{aligned} \{w_1, w_2\} &\not\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \\ \{w_1, w_3\} &\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \\ \{w_1, w_2, w_3\} &\not\models \mathrm{WasThere}\,(\mathrm{Luisa}) \to ?\,\mathrm{Guilty}\,(\mathrm{Luisa}) \end{aligned}$$

Syntax [Cia22]

Definition

- We call a set $\Sigma := (\mathsf{P}_\Sigma, \mathsf{F}_\Sigma, \mathrm{ar}_\Sigma, \mathrm{rigid}_\Sigma)$ a signature.
- P_Σ provides predicate symbols.
- F_Σ provides function symbols.
- $\mathrm{ar}_\Sigma \colon \mathsf{P}_\Sigma + \mathsf{F}_\Sigma \to \mathbb{N}$ maps symbols to their arity.
- $\mathrm{rigid}_\Sigma \subseteq \mathsf{F}_\Sigma$ indicates whether a function symbol is rigid.

Syntax [Cia22]

Definition

- We call a set $\Sigma := (\mathsf{P}_\Sigma, \mathsf{F}_\Sigma, \mathrm{ar}_\Sigma, \mathrm{rigid}_\Sigma)$ a signature.
- P_Σ provides predicate symbols.
- F_Σ provides function symbols.
- $\mathrm{ar}_\Sigma \colon \mathsf{P}_\Sigma + \mathsf{F}_\Sigma \to \mathbb{N}$ maps symbols to their arity.
- $\mathrm{rigid}_\Sigma \subseteq \mathsf{F}_\Sigma$ indicates whether a function symbol is rigid.

Fix a set Var of variables.

Syntax [Cia22]

Definition

- We call a set $\Sigma := (\mathsf{P}_\Sigma, \mathsf{F}_\Sigma, \mathrm{ar}_\Sigma, \mathrm{rigid}_\Sigma)$ a signature.
- P_Σ provides predicate symbols.
- F_Σ provides function symbols.
- $\mathrm{ar}_\Sigma \colon \mathsf{P}_\Sigma + \mathsf{F}_\Sigma \to \mathbb{N}$ maps symbols to their arity.
- $\mathrm{rigid}_\Sigma \subseteq \mathsf{F}_\Sigma$ indicates whether a function symbol is rigid.

Fix a set Var of variables.

Definition

Terms and Formulae over a signature Σ are defined as follows:

$$\begin{aligned} t \in \mathrm{Ter}_\Sigma &::= x \mid f\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(f)}\right) && f \in \mathsf{F}_\Sigma \\ \phi, \psi \in \mathcal{F}_\Sigma &::= P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) \mid \bot \mid \phi \to \psi \mid \phi \wedge \psi \mid \phi \mathbin{\mathbb{V}} \psi \mid \forall x.\, \phi \mid \exists x.\, \phi && P \in \mathsf{P}_\Sigma \\ ?\phi &:= \phi \mathbin{\mathbb{V}} \neg\phi \end{aligned}$$

Syntax
Regarding Rocq

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$\mathrm{Var} := \mathbb{N}$$
$$\phi \in \mathcal{F}_\Sigma ::= \ \dots \mid \forall.\,\phi \mid \exists\,.\,\phi$$

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$
\begin{aligned}
\mathrm{Var} &:= \mathbb{N} \\
\phi \in \mathcal{F}_\Sigma &::= \ \ldots \mid \forall.\,\phi \mid \exists\,.\,\phi
\end{aligned}
$$

- Use Autosubst [STS] library to implement substitutions.

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$\mathrm{Var} := \mathbb{N}$$
$$\phi \in \mathcal{F}_\Sigma ::= \ \dots \mid \forall .\, \phi \mid \exists \,.\, \phi$$

- Use Autosubst [STS] library to implement substitutions.
- Use Typeclasses to use signatures implicitly.

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$\mathrm{Var} := \mathbb{N}$$
$$\phi \in \mathcal{F}_\Sigma ::= \; \ldots \mid \forall.\, \phi \mid \exists\, .\, \phi$$

- Use Autosubst [STS] library to implement substitutions.
- Use Typeclasses to use signatures implicitly.
- Ensure decidable equality for predicate and function symbols to distinguish them.

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$\mathrm{Var} := \mathbb{N}$$
$$\phi \in \mathcal{F}_\Sigma ::= \; \ldots \mid \forall.\, \phi \mid \exists\, .\, \phi$$

- Use Autosubst [STS] library to implement substitutions.
- Use Typeclasses to use signatures implicitly.

- Ensure decidable equality for predicate and function symbols to distinguish them.
- Implement arities via arity types:

$$\mathrm{ar}_\Sigma \colon \mathsf{P}_\Sigma + \mathsf{F}_\Sigma \to |\mathbf{FinSet}|$$

Syntax
Regarding Rocq

- Implement variables via De Bruijn indices [dBr72]:

$$\mathrm{Var} := \mathbb{N}$$
$$\phi \in \mathcal{F}_\Sigma ::= \ \ldots \mid \forall . \, \phi \mid \exists \, . \, \phi$$

- Use Autosubst [STS] library to implement substitutions.
- Use Typeclasses to use signatures implicitly.

- Ensure decidable equality for predicate and function symbols to distinguish them.
- Implement arities via arity types:

$$\mathrm{ar}_\Sigma \colon \mathsf{P}_\Sigma + \mathsf{F}_\Sigma \to |\mathbf{FinSet}|$$

- Implement arguments via argument functions:

$$\begin{array}{llll} t ::= \ldots \mid f\,(args) & \text{where } args & : \mathrm{ar}_\Sigma\,(f) & \to \mathrm{Ter}_\Sigma \\ \phi ::= P\,(args) \mid \ldots & \text{where } args & : \mathrm{ar}_\Sigma\,(P) & \to \mathrm{Ter}_\Sigma \end{array}$$

Syntax

Regarding Rocq (2)

```
Class Signature :=
  {
    PSymb : Type;
    PSymb_EqDec :: EqDec (eq_setoid PSymb);
    PAri : PSymb → Type;
    FSymb : Type;
    FSymb_EqDec :: EqDec (eq_setoid FSymb);
    FAri : FSymb → Type;
    rigid : FSymb → bool
    (* ... *)
  }.

Inductive form `{Signature} :=
  | Pred : forall (p : PSymb), (PAri p → term) → form
  | Bot : var → form
  | Impl : form → form → form
  | Conj : form → form → form
  | Idisj : form → form → form
  | Forall : {bind term in form} → form
  | Iexists : {bind term in form} → form.
```

Syntax

Regarding Rocq (2)

```
Class Signature :=
  {
    PSymb : Type;
    PSymb_EqDec :: EqDec (eq_setoid PSymb);
    PAri : PSymb → Type;
    FSymb : Type;
    FSymb_EqDec :: EqDec (eq_setoid FSymb);
    FAri : FSymb → Type;
    rigid : FSymb → bool
    (* ... *)
  }.

Inductive form `{Signature} :=
  | Pred : forall (p : PSymb), (PAri p → term) → form
  | Bot : var → form
  | Impl : form → form → form
  | Conj : form → form → form
  | Idisj : form → form → form
  | Forall : {bind term in form} → form
  | Iexists : {bind term in form} → form.
```

- (Decidable) syntactic equality for formulae (and terms) becomes non-trivial because of dependent types.

Syntax

Regarding Rocq (2)

```
Class Signature :=
  {
    PSymb : Type;
    PSymb_EqDec :: EqDec (eq_setoid PSymb);
    PAri : PSymb → Type;
    FSymb : Type;
    FSymb_EqDec :: EqDec (eq_setoid FSymb);
    FAri : FSymb → Type;
    rigid : FSymb → bool
    (* ... *)
  }.

Inductive form `{Signature} :=
  | Pred : forall (p : PSymb), (PAri p → term) → form
  | Bot : var → form
  | Impl : form → form → form
  | Conj : form → form → form
  | Idisj : form → form → form
  | Forall : {bind term in form} → form
  | Iexists : {bind term in form} → form.
```

- (Decidable) syntactic equality for formulae (and terms) becomes non-trivial because of dependent types.
- Solution: Define a setoid equality for terms and formulae.

Syntax
Regarding Rocq (3)

```
Fixpoint term_eq `{S : Signature} (t : term) : term → Prop :=
  match t with
  | Var x1 ⇒
      fun t2 ⇒
      match t2 with
      | Var x2 ⇒ (x1 == x2)%type
      | _ ⇒ False
      end
  | Func f1 args1 ⇒
      fun t2 ⇒
      match t2 with
      | Func f2 args2 ⇒
          match equiv_dec f1 f2 with
          | left Heq ⇒
              term_eq_Func_Func_EqDec term_eq f1 args1 f2 args2 Heq
          | _ ⇒ False
          end
      | _ ⇒ False
      end
  end.
```

```
Definition term_eq_Func_Func_EqDec
  `{ S : Signature}
  (rec : relation term)
  (f1 : FSymb)
  (args1 : FAri f1 → term)
  (f2 : FSymb)
  (args2 : FAri f2 → term)
  (is_equal : (f1 == f2)%type) : Prop :=

  eq_rect
  f1
  (fun f ⇒ (FAri f → term) → Prop)
  (fun args ⇒
    forall arg,
      rec (args1 arg) (args arg)
  )
  f2
  is_equal
  args2.
```

Semantics
Models, States

Definition

Let Σ be a signature.

- A tuple $\mathfrak{M} := \left(\mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, (\mathfrak{M}_w\, [\![f]\!])_{w\in W, f\in \mathsf{F}_\Sigma}, (\mathfrak{M}_w\, [\![P]\!])_{w\in W, P\in \mathsf{P}_\Sigma}\right)$ is called a model.

Semantics
Models, States

Definition
Let Σ be a signature.

- A tuple $\mathfrak{M} := \left(\mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, (\mathfrak{M}_w \llbracket f \rrbracket)_{w \in W, f \in \mathsf{F}_\Sigma}, (\mathfrak{M}_w \llbracket P \rrbracket)_{w \in W, P \in \mathsf{P}_\Sigma}\right)$ is called a model.
- $\mathrm{W}_{\mathfrak{M}}$ is a set of possible worlds.

Semantics
Models, States

Definition

Let Σ be a signature.

- A tuple $\mathfrak{M} := \left(\mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, (\mathfrak{M}_w \llbracket f \rrbracket)_{w \in W, f \in \mathsf{F}_\Sigma}, (\mathfrak{M}_w \llbracket P \rrbracket)_{w \in W, P \in \mathsf{P}_\Sigma}\right)$ is called a model.
- $\mathrm{W}_{\mathfrak{M}}$ is a set of possible worlds.
- $\mathrm{I}_{\mathfrak{M}}$ is a (non-empty) set of individuals.
- $\mathfrak{M}_w \llbracket f \rrbracket : \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(f)} \to \mathrm{I}_{\mathfrak{M}}$ is the interpretation of f in a world w.
- $\mathfrak{M}_w \llbracket P \rrbracket \subseteq \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(P)}$ is the interpretation of P in a world w.

Semantics
Models, States

Definition

Let Σ be a signature.

- A tuple $\mathfrak{M} := \left(\mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, (\mathfrak{M}_w \llbracket f \rrbracket)_{w \in W, f \in \mathsf{F}_\Sigma}, (\mathfrak{M}_w \llbracket P \rrbracket)_{w \in W, P \in \mathsf{P}_\Sigma}\right)$ is called a model.
- $\mathrm{W}_{\mathfrak{M}}$ is a set of possible worlds.
- $\mathrm{I}_{\mathfrak{M}}$ is a (non-empty) set of individuals.
- $\mathfrak{M}_w \llbracket f \rrbracket : \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(f)} \to \mathrm{I}_{\mathfrak{M}}$ is the interpretation of f in a world w.
- $\mathfrak{M}_w \llbracket P \rrbracket \subseteq \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(P)}$ is the interpretation of P in a world w.
- for every rigid $f \in \mathsf{F}_\Sigma$ and for all $w_1, w_2 \in \mathrm{W}_{\mathfrak{M}}$ we have $\mathfrak{M}_{w_1} \llbracket f \rrbracket = \mathfrak{M}_{w_2} \llbracket f \rrbracket$.

Semantics

Models, States

Definition

Let Σ be a signature.

- A tuple $\mathfrak{M} := \left(\mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, (\mathfrak{M}_w [\![f]\!])_{w \in W, f \in \mathsf{F}_\Sigma}, (\mathfrak{M}_w [\![P]\!])_{w \in W, P \in \mathsf{P}_\Sigma}\right)$ is called a model.
- $\mathrm{W}_{\mathfrak{M}}$ is a set of possible worlds.
- $\mathrm{I}_{\mathfrak{M}}$ is a (non-empty) set of individuals.
- $\mathfrak{M}_w [\![f]\!] : \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(f)} \to \mathrm{I}_{\mathfrak{M}}$ is the interpretation of f in a world w.
- $\mathfrak{M}_w [\![P]\!] \subseteq \mathrm{I}_{\mathfrak{M}}^{\mathrm{ar}_\Sigma(P)}$ is the interpretation of P in a world w.
- for every rigid $f \in \mathsf{F}_\Sigma$ and for all $w_1, w_2 \in \mathrm{W}_{\mathfrak{M}}$ we have $\mathfrak{M}_{w_1} [\![f]\!] = \mathfrak{M}_{w_2} [\![f]\!]$.

Definition

Let Σ be a signature, $\mathfrak{M}$ be a model. A subset $s \subseteq \mathrm{W}_{\mathfrak{M}}$ is called an (information) state.

What should states be from Coq's point of view?? Boolean predicates of type `World` $\to$ `bool` (decidable by definition), or rather arbitrary functions of type `World` $\to$ `Prop`?

Semantics
Referent of a Term

Definition

Let Σ be a signature, $\mathfrak{M}$ be a Model, $s \subseteq \mathrm{W}_{\mathfrak{M}}$ an information state and $\eta\colon\ \mathrm{Var} \to \mathrm{I}_{\mathfrak{M}}$ a variable assignment. The referent of a term $t \in \mathrm{Ter}_{\Sigma}$ is defined as follows:

$$\mathfrak{M}_{w,\eta}\,[\![x]\!] := \eta\,(x)$$
$$\mathfrak{M}_{w,\eta}\left[\!\left[f\left(t_1,\ldots,t_{\mathrm{ar}_\Sigma(f)}\right)\right]\!\right] := \mathfrak{M}_{w}\,[\![f]\!]\left(\mathfrak{M}_{w,\eta}\,[\![t_1]\!]\,,\ldots,\mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_\Sigma(f)}\right]\!\right]\right)$$

Semantics
Referent of a Term

Definition

Let Σ be a signature, $\mathfrak{M}$ be a Model, $s \subseteq \mathrm{W}_{\mathfrak{M}}$ an information state and $\eta \colon \mathrm{Var} \to \mathrm{I}_{\mathfrak{M}}$ a variable assignment. The referent of a term $t \in \mathrm{Ter}_{\Sigma}$ is defined as follows:

$$\begin{aligned}\mathfrak{M}_{w,\eta}\left[\!\left[x\right]\!\right] &:= \eta\left(x\right)\\ \mathfrak{M}_{w,\eta}\left[\!\left[f\left(t_1,\ldots,t_{\mathrm{ar}_{\Sigma}(f)}\right)\right]\!\right] &:= \mathfrak{M}_{w}\left[\!\left[f\right]\!\right]\left(\mathfrak{M}_{w,\eta}\left[\!\left[t_1\right]\!\right],\ldots,\mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_{\Sigma}(f)}\right]\!\right]\right)\end{aligned}$$

Using the new syntax:

$$\mathfrak{M}_{w,\eta}\left[\!\left[f\left(args\right)\right]\!\right] := \mathfrak{M}_{w}\left[\!\left[f\right]\!\right]\left(\mathfrak{M}_{w,\eta}\left[\!\left[-\right]\!\right] \circ args\right)$$

Semantics
Support

Definition
The support relation $\models$ is defined as follows:

Semantics
Support

Definition

The support relation $\models$ is defined as follows:

$$\mathfrak{M}, s, \eta \models P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) :\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\left[\!\left[t_1\right]\!\right], \ldots, \mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_\Sigma(P)}\right]\!\right]\right) \in \mathfrak{M}_w\left[\!\left[P\right]\!\right]$$

Semantics
Support

Definition

The support relation $\models$ is defined as follows:

$$\begin{aligned}\mathfrak{M}, s, \eta \models P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) &:\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\left[\!\left[t_1\right]\!\right], \ldots, \mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_\Sigma(P)}\right]\!\right]\right) \in \mathfrak{M}_w\left[\!\left[P\right]\!\right] \\ \mathfrak{M}, s, \eta \models \bot &:\Longleftrightarrow s = \emptyset\end{aligned}$$

Semantics
Support

Definition

The support relation $\models$ is defined as follows:

$$\begin{aligned}
\mathfrak{M}, s, \eta \models P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) &:\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\left[\!\left[t_1\right]\!\right], \ldots, \mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_\Sigma(P)}\right]\!\right]\right) \in \mathfrak{M}_w\left[\!\left[P\right]\!\right] \\
\mathfrak{M}, s, \eta \models \bot &:\Longleftrightarrow s = \emptyset \\
\mathfrak{M}, s, \eta \models \phi \to \psi &:\Longleftrightarrow \text{for all } t \subseteq s,\ \mathfrak{M}, t, \eta \models \phi \text{ implies } \mathfrak{M}, t, \eta \models \psi
\end{aligned}$$

Semantics
Support

Definition

The support relation $\models$ is defined as follows:

$$\begin{aligned}
\mathfrak{M}, s, \eta \models P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) &:\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\left[\!\left[t_1\right]\!\right], \ldots, \mathfrak{M}_{w,\eta}\left[\!\left[t_{\mathrm{ar}_\Sigma(P)}\right]\!\right]\right) \in \mathfrak{M}_w\left[\!\left[P\right]\!\right] \\
\mathfrak{M}, s, \eta \models \bot &:\Longleftrightarrow s = \emptyset \\
\mathfrak{M}, s, \eta \models \phi \to \psi &:\Longleftrightarrow \text{for all } t \subseteq s,\ \mathfrak{M}, t, \eta \models \phi \text{ implies } \mathfrak{M}, t, \eta \models \psi \\
\mathfrak{M}, s, \eta \models \phi \wedge \psi &:\Longleftrightarrow \mathfrak{M}, s, \eta \models \phi \text{ and } \mathfrak{M}, s, \eta \models \psi \\
\mathfrak{M}, s, \eta \models \phi \vee\!\!\!\vee \psi &:\Longleftrightarrow \mathfrak{M}, s, \eta \models \phi \text{ or } \mathfrak{M}, s, \eta \models \psi
\end{aligned}$$

Semantics
Support

Definition

The support relation $\models$ is defined as follows:

$$
\begin{aligned}
\mathfrak{M}, s, \eta \models P\left(t_1, \ldots, t_{\mathrm{ar}_\Sigma(P)}\right) &:\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\llbracket t_1 \rrbracket, \ldots, \mathfrak{M}_{w,\eta}\left\llbracket t_{\mathrm{ar}_\Sigma(P)}\right\rrbracket\right) \in \mathfrak{M}_w\llbracket P \rrbracket \\
\mathfrak{M}, s, \eta \models \bot &:\Longleftrightarrow s = \emptyset \\
\mathfrak{M}, s, \eta \models \phi \to \psi &:\Longleftrightarrow \text{for all } t \subseteq s,\ \mathfrak{M}, t, \eta \models \phi \text{ implies } \mathfrak{M}, t, \eta \models \psi \\
\mathfrak{M}, s, \eta \models \phi \wedge \psi &:\Longleftrightarrow \mathfrak{M}, s, \eta \models \phi \text{ and } \mathfrak{M}, s, \eta \models \psi \\
\mathfrak{M}, s, \eta \models \phi \mathbin{\rotatebox[origin=c]{180}{$\wedge$}\!\!\vee} \psi &:\Longleftrightarrow \mathfrak{M}, s, \eta \models \phi \text{ or } \mathfrak{M}, s, \eta \models \psi \\
\mathfrak{M}, s, \eta \models \forall x.\, \phi &:\Longleftrightarrow \text{for all } i \in \mathrm{I}_{\mathfrak{M}},\ \mathfrak{M}, s, \eta[x \mapsto i] \models \phi \\
\mathfrak{M}, s, \eta \models \exists x.\, \phi &:\Longleftrightarrow \text{there exists } i \in \mathrm{I}_{\mathfrak{M}},\ \mathfrak{M}, s, \eta[x \mapsto i] \models \phi
\end{aligned}
$$

Using the new syntax:

$$
\begin{aligned}
\mathfrak{M}, s, \eta \models P\,(args) &:\Longleftrightarrow \text{for all } w \in s \text{ we have } \left(\mathfrak{M}_{w,\eta}\llbracket - \rrbracket \circ args\right) \in \mathfrak{M}_w\llbracket P \rrbracket \\
\mathfrak{M}, s, \eta \models \forall.\, \phi &:\Longleftrightarrow \text{for all } i \in \mathrm{I}_{\mathfrak{M}},\ \mathfrak{M}, s, i \bullet \eta \models \phi \\
\mathfrak{M}, s, \eta \models \exists.\, \phi &:\Longleftrightarrow \text{there exists } i \in \mathrm{I}_{\mathfrak{M}},\ \mathfrak{M}, s, i \bullet \eta \models \phi
\end{aligned}
$$

Semantics
Various properties

Persistency

$t \subseteq s$ and $\mathfrak{M}, s, \eta \models \phi \Longrightarrow \mathfrak{M}, t, \eta, \models \phi$

Empty State Property

$$\mathfrak{M}, \emptyset, \eta \models \phi$$

Semantics
Various properties

Persistency

$t \subseteq s$ and $\mathfrak{M}, s, \eta \models \phi \Longrightarrow \mathfrak{M}, t, \eta, \models \phi$

Empty State Property

$$\mathfrak{M}, \emptyset, \eta \models \phi$$

- $\mathfrak{M}|_s := (s \subseteq \mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, \ldots)$

Locality

$$\mathfrak{M}, s, \eta \models \phi \Longleftrightarrow \mathfrak{M}|_s, s, \eta \models \phi$$

Semantics

Various properties

Persistency

$t \subseteq s$ and $\mathfrak{M}, s, \eta \models \phi \Longrightarrow \mathfrak{M}, t, \eta, \models \phi$

Empty State Property

$$\mathfrak{M}, \emptyset, \eta \models \phi$$

- $\mathfrak{M}|_s := (s \subseteq \mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, \ldots)$

Locality

$$\mathfrak{M}, s, \eta \models \phi \Longleftrightarrow \mathfrak{M}|_s, s, \eta \models \phi$$

- Defining $\mathfrak{M}|_s$ in Rocq needs subtypes.
- Solution: Generalize $\mathrm{W}_{\mathfrak{M}}$ to a setoid.

Semantics

Various properties

Persistency

$t \subseteq s$ and $\mathfrak{M}, s, \eta \models \phi \Longrightarrow \mathfrak{M}, t, \eta, \models \phi$

Empty State Property

$$\mathfrak{M}, \emptyset, \eta \models \phi$$

- $\mathfrak{M}|_s := (s \subseteq \mathrm{W}_{\mathfrak{M}}, \mathrm{I}_{\mathfrak{M}}, \ldots)$

Locality

$$\mathfrak{M}, s, \eta \models \phi \Longleftrightarrow \mathfrak{M}|_s, s, \eta \models \phi$$

- Defining $\mathfrak{M}|_s$ in Rocq needs subtypes.
- Solution: Generalize $\mathrm{W}_{\mathfrak{M}}$ to a setoid.

```
Context `{M : Model}. Context (s : state).

Program Definition restricted_Model : Model :=
  {|
    World := {w : World | contains s w};
    World_Setoid := sig_Setoid (contains_Morph s);
    PInterpretation w := PInterpretation (proj1_sig w);
    FInterpretation w := FInterpretation (proj1_sig w);
    (* ... *)
  |}.

Program Definition restricted_state (t : state) :
  @state _ (restricted_Model s) := (* ... *)

Program Definition unrestricted_state
  (t : @state _ (restricted_Model s)) : state := (* ... *)

Proposition locality `{M : Model} :
  forall phi s a t, substate t s →
    support phi t a ↔ support phi (@restricted_state _ M s t) a.
```

Semantics

InqFOL

Definition

Define Inquisitive First-Order Logic as follows:

$$\mathsf{InqLog}_\Sigma := \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M}, s \subseteq \mathrm{W}_\mathfrak{M}, \eta\colon \mathrm{Var} \to \mathrm{I}_\mathfrak{M}\}$$

Semantics
InqFOL

Definition

Define Inquisitive First-Order Logic as follows:

$$\text{InqLog}_\Sigma := \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M}, s \subseteq \mathrm{W}_\mathfrak{M}, \eta\colon \ \mathrm{Var} \to \mathrm{I}_\mathfrak{M}\}$$

- There exists a ND-System by Ciardelli/Grilletti [CG22] which is sound, but not yet proven to be complete.

Boundedness

Introduction

- Restricting the set of worlds to be finite yields Bounded Inquisitive FOL.

$$\mathsf{InqLogB}_{\Sigma,n} := \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M} \text{ with } |\mathrm{W}_{\mathfrak{M}}| < n, s \subseteq \mathrm{W}_{\mathfrak{M}}, \eta\colon \mathrm{Var} \to \mathrm{I}_{\mathfrak{M}}\}$$

$$\mathsf{InqLogB}_{\Sigma} := \bigcap_{n\in\mathbb{N}} \mathsf{InqLogB}_{\Sigma,n}$$

$$= \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M}, s \subseteq_{\mathsf{fin}} \mathrm{W}_{\mathfrak{M}}, \eta\colon \mathrm{Var} \to \mathrm{I}_{\mathfrak{M}}\}$$

Boundedness

Introduction

- Restricting the set of worlds to be finite yields Bounded Inquisitive FOL.

$$\mathsf{InqLogB}_{\Sigma,n} := \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M} \text{ with } |\mathrm{W}_\mathfrak{M}| < n, s \subseteq \mathrm{W}_\mathfrak{M}, \eta\colon \mathrm{Var} \to \mathrm{I}_\mathfrak{M}\}$$

$$\mathsf{InqLogB}_{\Sigma} := \bigcap_{n\in\mathbb{N}} \mathsf{InqLogB}_{\Sigma,n}$$

$$= \{\phi \in \mathcal{F}_\Sigma \mid \mathfrak{M}, s, \eta \models \phi \text{ for all models } \mathfrak{M}, s \subseteq_{\mathsf{fin}} \mathrm{W}_\mathfrak{M}, \eta\colon \mathrm{Var} \to \mathrm{I}_\mathfrak{M}\}$$

- Ciardelli/Griletti [CG22] extended their ND-System for $\mathsf{InqLogB}_{\Sigma,n}$ and it proved the resulting extensions to be complete *(for most signatures)*.
- Added axiom: Cardinality Formula, which depends on the concrete signature.
- Apart from signature-dependency, such axioms seem to destroy most desirable proof-theoretic properties of a ND system . . .

Split rules of the ND system of Ciardelli & Grilletti

$$\frac{\alpha \to \varphi \mathbin{\vee\!\!\!\vee} \psi}{(\alpha \to \varphi) \mathbin{\vee\!\!\!\vee} (\alpha \to \psi)}$$

$$\frac{\alpha \to \exists x.\psi \qquad\qquad x \notin \mathrm{FV}(\alpha)}{\exists x.\alpha \to \varphi}$$

Figure 1: Split rules

- Their soundness [Cia15, Proposition 4.4.6] relies on the definition of the state

$$|\alpha|_{\mathfrak{M}} := \{w \in \mathrm{W}_{\mathfrak{M}} \mid \mathfrak{M}, w \models_{\eta} \alpha\}$$

  for a classical formula α and a model $\mathfrak{M}$.
- It is easy to show by a reduction from classical first-order logic that $\models$ is undecidable.
- Therefore, we cannot use boolean predicates to represent states in order to formalize the soundness proof for this natural deduction system as we would not even be able to define $|\alpha|_{\mathfrak{M}}$.

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_{\Sigma}$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_{\Sigma}$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity
- Labels: Finite sets of natural numbers.

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_{\Sigma}$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity
- Labels: Finite sets of natural numbers.
- Sequents: $\Gamma \Rightarrow \Delta$ where Γ, Δ are finite sets of labelled formulae such as $(\{1, 2\}, \phi)$.

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_{\Sigma}$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity
- Labels: Finite sets of natural numbers.
- Sequents: $\Gamma \Rightarrow \Delta$ where Γ, Δ are finite sets of labelled formulae such as $(\{1,2\}, \phi)$.
- Semantics of a labelled formula (X, ϕ) are given by a mapping $f : \mathbb{N} \to \mathrm{W}_{\mathfrak{M}}$.

- Semantics of a sequent $\Gamma \Rightarrow \Delta$:

$$\begin{array}{rll} \text{If } \mathfrak{M}, f, \eta \models (X, \phi) & \text{for all} & (X, \phi) \in \Gamma, \\ \text{then } \mathfrak{M}, f, \eta \models (X, \psi) & \text{for some} & (Y, \phi) \in \Delta \end{array}$$

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_\Sigma$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity
- Labels: Finite sets of natural numbers.
- Sequents: $\Gamma \Rightarrow \Delta$ where Γ, Δ are finite sets of labelled formulae such as $(\{1,2\}, \phi)$.
- Semantics of a labelled formula (X, ϕ) are given by a mapping $f : \mathbb{N} \to \mathrm{W}_{\mathfrak{M}}$.

- Semantics of a sequent $\Gamma \Rightarrow \Delta$:

$$\begin{array}{rll} \text{If } \mathfrak{M}, f, \eta \models (X, \phi) & \text{for all} & (X, \phi) \in \Gamma, \\ \text{then } \mathfrak{M}, f, \eta \models (X, \psi) & \text{for some} & (Y, \phi) \in \Delta \end{array}$$

- The TABLEAUX setup for simplicity restricted to purely relational signatures; no function symbols, no complex terms, no discussion of rigidity

A Sequent Calculus

- L./Sano [LS25] provide a sequent calculus for $\mathsf{InqLogB}_\Sigma$ which is proved to be sound and complete for each $\mathsf{InqLogB}_{\Sigma,n}$ (with a corresponding restriction on labels)
- And the calculus provides at least a sufficient characterization of schematic validity
- Labels: Finite sets of natural numbers.
- Sequents: $\Gamma \Rightarrow \Delta$ where Γ, Δ are finite sets of labelled formulae such as $(\{1,2\}, \phi)$.
- Semantics of a labelled formula (X, ϕ) are given by a mapping $f : \mathbb{N} \to \mathrm{W}_{\mathfrak{M}}$.
- Semantics of a sequent $\Gamma \Rightarrow \Delta$:

$$\begin{array}{rll} \text{If } \mathfrak{M}, f, \eta \models (X, \phi) & \text{for all} & (X, \phi) \in \Gamma, \\ \text{then } \mathfrak{M}, f, \eta \models (X, \psi) & \text{for some} & (Y, \phi) \in \Delta \end{array}$$

- The TABLEAUX setup for simplicity restricted to purely relational signatures; no function symbols, no complex terms, no discussion of rigidity
- MO Elliger slightly adapted the sequent calculus of L./Sano to cover , e.g., nontrivial rigid terms.

A Sequent Calculus

Some Rules

$$\frac{(\emptyset, \phi) \in \Delta}{\Gamma \Rightarrow \Delta}\text{(empty)}$$

$$\frac{(X, \bot) \in \Gamma \quad n \in X}{\Gamma \Rightarrow \Delta}(\bot \Rightarrow)$$

$$\frac{(X, \phi \to \psi) \in \Delta \quad \{\Gamma, (Y, \phi) \Rightarrow (Y, \psi), \Delta \mid Y \subseteq X\}}{\Gamma \Rightarrow \Delta}(\Rightarrow\to)$$

$$\frac{(X, \phi \mathbin{\vee\!\!\!\vee} \psi) \in \Delta \quad \Gamma \Rightarrow (X, \phi), (X, \psi), \Delta}{\Gamma \Rightarrow \Delta}(\Rightarrow \mathbin{\vee\!\!\!\vee})$$

$$\frac{(X, \exists\,.\,\phi) \in \Delta \quad t \text{ is rigid} \quad \Gamma \Rightarrow (X, \phi.\,[t \bullet \mathrm{ids}]), \Delta}{\Gamma \Rightarrow \Delta}(\Rightarrow \exists)$$

$$\frac{(X, \phi \mathbin{\vee\!\!\!\vee} \psi) \in \Gamma \quad \Gamma, (X, \phi) \Rightarrow \Delta \quad \Gamma, (X, \psi) \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}(\mathbin{\vee\!\!\!\vee} \Rightarrow)$$

$$\frac{(X, \exists\,.\,\phi) \in \Gamma \quad \Gamma.\,[(+1)], (X, \phi) \Rightarrow \Delta.\,[(+1)]}{\Gamma \Rightarrow \Delta}(\exists \Rightarrow)$$

Sequent Calculus

Some Notes

- The rule of cut is proven to be admissible by L./Sano.
- Inside our formalization, we hardcoded it without showing admissibility.
- Our implementation of the sequent calculus also comes with a proof of soundness.
- The implementation with its extended syntax and rules currently lacks a (mechanized) proof of completeness.

```
Inductive Seq `{Signature} : relation (list lb_form) :=
  (* ... *)
  | Seq_Iexists_r :
      forall ls rs ns phi t,
        InS (pair ns <{iexists phi}>) rs →
        term_rigid t →
        Seq ls ((pair ns phi.|[t/]) :: rs) →
        Seq ls rs.

Theorem soundness `{Signature} :
  forall Phi Psi, Seq Phi Psi →
    satisfaction_conseq Phi Psi.
Proof.
  induction 1. (* on Seq Phi Psi *)
  all: eauto using
    satisfaction_conseq_empty,
    satisfaction_conseq_id,
    (* ... *).
Qed.
```

- Labels are implemented via lists in a suitable way
- Proof of soundness not only allows, but naturally seems to require a decidable notion of state!

Conclusions & Future Work

Develop this further! Perhaps so that both ND and sequent system can be handled in an uniform setting?

References

References

[CG22] I. Ciardelli and G. Grilletti. "Coherence in inquisitive first-order logic". en. In: Annals of Pure and Applied Logic 173.9 (Oct. 2022), p. 103155. DOI: 10.1016/j.apal.2022.103155.

[Cia15] I. A. Ciardelli. "Questions in Logic". en. doctoral. University of Amsterdam, Dec. 2015.

[Cia22] I. Ciardelli. Inquisitive Logic: Consequence and Inference in the Realm of Questions. en. Vol. 60. Trends in Logic. Cham: Springer International Publishing, 2022. DOI: 10.1007/978-3-031-09706-5.

[dBr72] N. G de Bruijn. "Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem". In: Indagationes Mathematicae (Proceedings) 75.5 (Jan. 1972), pp. 381–392. DOI: 10.1016/1385-7258(72)90034-0.

[LS25] T. Litak and K. Sano. Bounded Inquisitive Logics: Sequent Calculi and Schematic Validity. en. A modified and expanded version of a paper accepted for TABLEAUX 2025. 2025.

[STS] S. Schäfer, T. Tebbi, and G. Smolka. "Autosubst: Reasoning with de Bruijn Terms and Parallel Substitution". en. In: ().