

Elementos de
lógica
formalizados
en
Isabelle/HOL

Sofía Santiago
Fernández

Objetivo:
motivación y
contexto del
problema

Desarrollo histórico
de la lógica

Lógica y
computación

Demostradores
interactivos: Isabelle

Desarrollo

Métodos en
Isabelle / HOL

Sintaxis

Semántica

Lema de Hintikka

Conclusiones

Elementos de lógica formalizados en Isabelle/HOL

Sofía Santiago Fernández

Departamento de Ciencias de la Computación e Inteligencia Artificial
Facultad de Matemáticas

Trabajo Fin de Grado

26 de junio de 2020



1 Objetivo: motivación y contexto del problema

- Desarrollo histórico de la lógica
- Lógica y computación
- Demostradores interactivos: Isabelle

2 Desarrollo

- Métodos en Isabelle/HOL
- Sintaxis
- Semántica
- Lema de Hintikka

3 Conclusiones

Sofía Santiago
Fernández

Objetivo:
motivación y
contexto del
problema

Desarrollo histórico
de la lógica

Lógica y
computación

Demostradores
interactivos: Isabelle

Desarrollo

Métodos en
Isabelle / HOL

Sintaxis

Semántica

Lema de Hintikka

Conclusiones

Formalización de elementos de la lógica proposicional en
Isabelle/HOL:

- 1 Sintaxis
- 2 Semántica
- 3 Lema de Hintikka

Influencias

- *Propositional Proof Systems* de Julius Michaelis y Tobias Nipkow
- *First – Order Logic and Automated Theorem Proving* de Melvin Fitting

Desarrollo histórico de la lógica

Origen: Antigua Grecia con Aristóteles.

s.XVII: Leibniz con su programa lógico que pretende:

- Sistema simbólico del lenguaje natural
- Matematización del concepto de validez

Lógica moderna

Frege instaura el tratamiento sistemático de la **lógica proposicional** basado en:

- Sintaxis completa
- Semántica asociada

Pertenece a la escuela del **logicismo**, que busca la formalización de las matemáticas.

Desarrollo de la computación y la IA \implies Permite formalizar las matemáticas y la lógica en lenguaje computacional.

Razonamiento automático

Basado en el **programa de Leibniz**:

- 1 Formalización rigurosa
- 2 Algoritmos para el razonamiento

Aplicaciones de la lógica en la computación

- Verificación y síntesis de programas
- Representación del conocimiento y razonamiento

Demostradores interactivos: Isabelle

Isabelle (1986: Larry Paulson y Tobias Nipkow)

Formalización lógica + Herramientas de razonamiento automático

- **Verificación** del razonamiento.
- **Mejora de la productividad** en el proceso de demostración.
- **Extensa librería de resultados formalizados** que continúan en desarrollo. (*The Alexandria Project: Large – Scale Formal Proof for the Working Mathematician*).

Otros demostradores interactivos: Coq, ACL2, PVS

Isabelle/HOL: especificación de Isabelle para lógica de orden superior.

1 Objetivo: motivación y contexto del problema

- Desarrollo histórico de la lógica
- Lógica y computación
- Demostradores interactivos: Isabelle

2 Desarrollo

- Métodos en Isabelle/HOL
- Sintaxis
- Semántica
- Lema de Hintikka

3 Conclusiones

Método seguido en el trabajo

- 1 Presentación de los contenidos teóricos.
- 2 Formalización en Isabelle/HOL.

Tácticas de demostración empleadas en Isabelle/HOL:

Demostración detallada

- Completada con definiciones y reglas elementales de las librerías.
- Resultado de una búsqueda inversa.

Demostración automática

- Emplea todas las herramientas de razonamiento automático.
- Gran eficiencia.

Sintaxis – Fórmulas proposicionales

Variables proposicionales

Conectivas: \neg , \wedge , \vee , \rightarrow

Definición (Fórmulas proposicionales)

Se definen recursivamente como:

- Las variables proposicionales son fórmulas llamadas *atómicas*.
- \perp
- Si F y G son fórmulas $\Rightarrow \neg F$, $F \wedge G$, $F \vee G$ y $F \rightarrow G$ también.

Principio de inducción sobre la estructura \Rightarrow táctica *induct* en Isabelle/HOL.

```
datatype (atoms: 'a) formula =  
  Atom 'a  
  | Bot           (⊥)  
  | Not 'a formula   (¬)  
  | And 'a formula 'a formula  (infix ∧ 68)  
  | Or 'a formula 'a formula   (infix ∨ 68)  
  | Imp 'a formula 'a formula  (infixr → 68)
```

atoms obtiene el conjunto de átomos
de una fórmula (es finito)

lemma finite (atoms F)

Sintaxis – Subfórmulas (I)

Definición (Conjunto de subfórmulas de una fórmula)

Se define recursivamente como:

$$\text{Subf}(F) = \begin{cases} \{F\} & \text{si } F \text{ es atómica.} \\ \{F\} \cup \text{Subf}(G), & \text{si } F = \neg G. \\ \{F\} \cup \text{Subf}(G) \cup \text{Subf}(H), & \text{si } F \text{ es } G * H. \end{cases}$$

(donde * es \wedge , \vee o \rightarrow .)

```
primrec subformulae :: 'a formula ⇒ 'a formula list where
  subformulae (Atom k) = [Atom k]
  | subformulae ⊥      = [⊥]
  | subformulae (¬ F)  = (¬ F) # subformulae F
  | subformulae (F ∧ G) = (F ∧ G) # subformulae F @ subformulae G
  | subformulae (F ∨ G) = (F ∨ G) # subformulae F @ subformulae G
  | subformulae (F → G) = (F → G) # subformulae F @ subformulae G
```

```
abbreviation setSubformulae :: 'a formula ⇒ 'a formula set where
  setSubformulae F ≡ set (subformulae F)
```

setSubformulae devuelve el **conjunto** de subfórmulas de una fórmula.

Resultados destacados

● Lema (Propia pertenencia)

Toda fórmula es subfórmula de ella misma.

lemma $F \in setSubformulae F$

● Lema

Todas las fórmulas atómicas de una fórmula son subfórmulas.

lemma $Atom ` atoms F \subseteq setSubformulae F$

● Lema (Transitividad)

Sea H una subfórmula de G que es a su vez subfórmula de F , entonces H es subfórmula de F .

lemma

assumes $G \in setSubformulae F$

$H \in setSubformulae G$

shows $H \in setSubformulae F$

Sofía Santiago
Fernández

Objetivo:
motivación y
contexto del
problema

Desarrollo histórico
de la lógica

Lógica y
computación

Demostradores
interactivos: Isabelle

Desarrollo

Métodos en
Isabelle/HOL

Sintaxis

Semántica

Lema de Hintikka

Conclusiones

Definición (\top)

```
definition Top ( $\top$ ) where  
 $\top \equiv \perp \rightarrow \perp$ 
```

Def. (Conjunción generalizada)

```
primrec BigAnd :: "'a formula list ⇒ 'a formula ( $\wedge$ -) where  
   $\wedge [] = (\neg \perp)$   
  |  $\wedge (F#Fs) = F \wedge \wedge Fs$ 
```

Def. (Disyunción generalizada)

```
primrec BigOr :: "'a formula list ⇒ 'a formula ( $\vee$ -) where  
   $\vee [] = \perp$   
  |  $\vee (F#Fs) = F \vee \vee Fs$ 
```

● Lema

El conjunto de átomos de la conjunción generalizada de una lista de fórmulas es la unión de los conjuntos de átomos de cada fórmula de la lista.

```
lemma atoms ( $\wedge Fs$ ) =  $\bigcup$  (atoms ' set Fs)
```

Semántica – Valor de una fórmula en una interpretación

Definición (Interpretación): aplicación del conjunto de variables proposicionales en los booleanos.

type-synonym 'a valuation = 'a ⇒ bool

Def. (Valor de una fórmula en una interpretación)

Para cada interpretación I existe una aplicación I' desde el conjunto de fórmulas a los booleanos definida recursivamente sobre la estructura de las fórmulas: (notado $I \models F$)

- $I'(p) = I(p)$
- $I'(\perp) = \text{False}$
- $I'(\neg F) = \neg I'(F)$
- $I'(F \wedge G) = I'(F) \wedge I'(G)$
- $I'(F \vee G) = I'(F) \vee I'(G)$
- $I'(F \rightarrow G) = I'(F) \rightarrow I'(G)$

```
primrec formula-semantics ::  
  'a valuation ⇒ 'a formula ⇒ bool (infix |≡ 51) where  
  A |≡ Atom k = A k  
  | A |≡ ⊥ = False  
  | A |≡ Not F = (¬ A |≡ F)  
  | A |≡ And F G = (A |≡ F ∧ A |≡ G)  
  | A |≡ Or F G = (A |≡ F ∨ A |≡ G)  
  | A |≡ Imp F G = (A |≡ F → A |≡ G)
```

Resultados que caracterizan el valor de una fórmula en una interpretación

lemma $p \notin \text{atoms } F \implies (\mathcal{A}(p := V)) \models F \longleftrightarrow \mathcal{A} \models F$

lemma $\forall k \in \text{atoms } F. \mathcal{A}_1 k = \mathcal{A}_2 k \implies \mathcal{A}_1 \models F \longleftrightarrow \mathcal{A}_2 \models F$

Semántica – Nociones de modelo y satisfacibilidad

Definición (Modelo de una fórmula): Una interpretación es modelo de una fórmula si el valor de la fórmula en dicha interpretación es *Verdadero*.

definition $\text{isModel } \mathcal{A} F \equiv \mathcal{A} \models F$

Definición (Modelo de un conjunto de fórmulas): Interpretación que es modelo de todas las fórmulas del conjunto.

definition $\text{isModelSet } \mathcal{A} S \equiv \forall F. (F \in S \longrightarrow \mathcal{A} \models F)$

Definición (Fórmula satisfacible): Si tiene algún modelo.

definition $\text{satF}(F) \equiv \exists \mathcal{A}. \mathcal{A} \models F$

Definición (Conjunto de fórmulas satisfacibles): Si tiene algún modelo.

definition $\text{sat } S \equiv \exists \mathcal{A}. \forall F \in S. \mathcal{A} \models F$

● Lema (Caracterización de modelo de la conjunción generalizada de una lista de fórmulas)

Una interpretación es modelo de la conjunción generalizada de una lista de fórmulas si y solo si es modelo de cada fórmula de la lista.

$$\text{lemma } (\mathcal{A} \models \bigwedge F_s) \longleftrightarrow (\forall f \in \text{set } F_s. \mathcal{A} \models f)$$

● Lema (Caracterización de modelo de la disyunción generalizada de una lista de fórmulas)

Una interpretación es modelo de la disyunción generalizada de una lista de fórmulas si y solo si existe una fórmula en la lista de la cual sea modelo.

$$\text{lemma } (\mathcal{A} \models \bigvee F_s) \longleftrightarrow (\exists f \in \text{set } F_s. \mathcal{A} \models f)$$

Semántica – Tautología. Consecuencia lógica

Definición (Tautología): Fórmula cuyo valor es Verdadero en toda interpretación.

abbreviation valid ($\models - 51$) where
 $\models F \equiv \forall \mathcal{A}. \mathcal{A} \models F$

- **Lema:** \top es tautología.

lemma $\mathcal{A} \models \top$

Definición (Consecuencia lógica) Una fórmula es **consecuencia lógica** de un conjunto de fórmulas si todos los modelos del conjunto son modelos de la fórmula.

definition entailment ::

'a formula set ⇒ 'a formula ⇒ bool ((- \models / -)
[53,53] 53) where
 $\Gamma \models F \equiv (\forall \mathcal{A}. ((\forall G \in \Gamma. \mathcal{A} \models G) \longrightarrow (\mathcal{A} \models F)))$

- **Lema (Caracterización de conjunto insatisfacible)** \perp es consecuencia lógica de un conjunto si y solo si el conjunto es insatisfacible.

lemma $\Gamma \models \perp \longleftrightarrow \neg \text{sat } \Gamma$

Lema de Hintikka – Conjuntos de Hintikka

Def. (Conjunto de Hintikka)

Conjunto de fórmulas S que
verifique:

1. $\perp \notin S$.
2. $p \in S \implies \neg p \notin S$.
3. $F \wedge G \in S \implies F \in S \text{ y } G \in S$.
4. $F \vee G \in S \implies F \in S \text{ o } G \in S$.
5. $F \rightarrow G \in S \implies \neg F \in S \text{ o } G \in S$.
6. $\neg(\neg F) \in S \implies F \in S$.
7. $\neg(F \wedge G) \in S \implies \neg F \in S \text{ o } \neg G \in S$.
8. $\neg(F \vee G) \in S \implies \neg F \in S \text{ y } \neg G \in S$.
9. $\neg(F \rightarrow G) \in S \implies F \in S \text{ y } \neg G \in S$.

- **Lema:** Sea S conjunto de Hintikka. Una fórmula no pertenece a S si su negación sí pertenece al mismo.

definition Hintikka $S \equiv$

$$\begin{aligned} & (\perp \notin S) \\ & \wedge (\forall k. Atom k \in S \longrightarrow \neg(Atom k) \in S \longrightarrow False) \\ & \wedge (\forall F G. F \wedge G \in S \longrightarrow F \in S \wedge G \in S) \\ & \wedge (\forall F G. F \vee G \in S \longrightarrow F \in S \vee G \in S) \\ & \wedge (\forall F G. F \rightarrow G \in S \longrightarrow \neg F \in S \vee G \in S) \\ & \wedge (\forall F. \neg(\neg F) \in S \longrightarrow F \in S) \\ & \wedge (\forall F G. \neg(F \wedge G) \in S \longrightarrow \neg F \in S \vee \neg G \in S) \\ & \wedge (\forall F G. \neg(F \vee G) \in S \longrightarrow \neg F \in S \wedge \neg G \in S) \\ & \wedge (\forall F G. \neg(F \rightarrow G) \in S \longrightarrow F \in S \wedge \neg G \in S)) \end{aligned}$$

Cada condición de la definición se separa como condición necesaria de conjunto de Hintikka.

lemma Hintikka $S \implies (\neg F \in S \longrightarrow F \notin S)$

Lema de Hintikka – Interpretación asociada al conjunto

Definición (Interpretación asociada a un conjunto de fórmulas): aquella que devuelve *Verdadero* sobre las variables proposicionales cuya correspondiente fórmula atómica pertenece al conjunto, y *Falso* en caso contrario.

```
definition setValuation ::  
  ('a formula) set ⇒ 'a valuation where  
  setValuation S ≡ λk. Atom k ∈ S
```

● Lema (1)

La interpretación asociada a un conjunto de Hintikka es modelo de una fórmula si esta pertenece al conjunto, y no es modelo de una fórmula si su negación pertenece al conjunto.

```
lemma Hintikka S ⇒ (F ∈ S → isModel (setValuation S) F)  
  ∧ (¬ F ∈ S → (¬ (isModel (setValuation S) F)))
```

Lema de Hintikka – Esquema de demostración detallada en Isabelle: Lema (1)

Caso (semi)detallado correspondiente a las fórmulas atómicas

```
lemma Hintikka-lemma-l2:
  assumes Hintikka S
  shows (F ∈ S → isModel (setValuation S) F) ∧
        (¬ F ∈ S → ¬(isModel (setValuation S) F))
proof (induct F)
  fix x
  show (Atom x ∈ S → isModel (setValuation S) (Atom x)) ∧
    (¬(Atom x) ∈ S → ¬isModel (setValuation S) (Atom x))
    using assms by rule HI2-1
next
  show (⊥ ∈ S → isModel (setValuation S) ⊥) ∧
    (¬⊥ ∈ S → ¬isModel (setValuation S) ⊥)
    using assms by rule HI2-2
next
  fix F
  show (F ∈ S → isModel (setValuation S) F) ∧
    (¬ F ∈ S → ¬isModel (setValuation S) F) ==>
    (¬ F ∈ S → isModel (setValuation S) (¬ F)) ∧
    (¬(¬ F) ∈ S → ¬isModel (setValuation S) (¬ F))
    using assms by rule HI2-3
next
  fix F1 F2
  show (F1 ∈ S → isModel (setValuation S) F1) ∧
    (¬ F1 ∈ S → ¬isModel (setValuation S) F1) ==>
    (F2 ∈ S → isModel (setValuation S) F2) ∧
    (¬ F2 ∈ S → ¬isModel (setValuation S) F2) ==>
    (F1 ∨ F2 ∈ S → isModel (setValuation S) (F1 ∨ F2)) ∧
    (¬(F1 ∨ F2) ∈ S → ¬isModel (setValuation S) (F1 ∨ F2))
    using assms by rule HI2-4
next
  fix F1 F2
  show (F1 ∈ S → isModel (setValuation S) F1) ∧
    (¬ F1 ∈ S → ¬isModel (setValuation S) F1) ==>
    (F2 ∈ S → isModel (setValuation S) F2) ∧
    (¬ F2 ∈ S → ¬isModel (setValuation S) F2) ==>
    (F1 ∨ F2 ∈ S → isModel (setValuation S) (F1 ∨ F2)) ∧
    (¬(F1 ∨ F2) ∈ S → ¬isModel (setValuation S) (F1 ∨ F2))
    using assms by rule HI2-5
next
  fix F1 F2
  show (F1 ∈ S → isModel (setValuation S) F1) ∧
    (¬ F1 ∈ S → ¬isModel (setValuation S) F1) ==>
    (F2 ∈ S → isModel (setValuation S) F2) ∧
    (¬ F2 ∈ S → ¬isModel (setValuation S) F2) ==>
    (F1 → F2 ∈ S → isModel (setValuation S) (F1 → F2)) ∧
    (¬(F1 → F2) ∈ S → ¬isModel (setValuation S) (F1 → F2))
    using assms by rule HI2-6
qed
```

```
lemma
  assumes Hintikka S
  shows ∀x. (Atom x ∈ S → isModel (setValuation S) (Atom x)) ∧
            (¬(Atom x) ∈ S → ¬isModel (setValuation S) (Atom x))
proof
  show ∀x. Atom x ∈ S → isModel (setValuation S) (Atom x)
  proof
    fix x
    assume Atom x ∈ S
    thus isModel (setValuation S) (Atom x)
      by (simp add: setValuation-def isModel-def)
  qed
  next
  show
    ∀x. ¬(Atom x) ∈ S → ¬isModel (setValuation S) (Atom x)
    proof
      fix x
      assume ¬(Atom x) ∈ S
      hence Atom x ∉ S using assms
      by (simp add: (¬(Atom x)) ∈ S Hintikka-l10)
      thus ¬isModel (setValuation S) (Atom x)
        by (simp add: isModel-def setValuation-def)
    qed
  qed
```

```
fix x
assume ¬(Atom x) ∈ S
hence Atom x ∉ S using assms
by (simp add: (¬(Atom x)) ∈ S Hintikka-l10)
thus ¬isModel (setValuation S) (Atom x)
  by (simp add: isModel-def setValuation-def)
qed
qed
```

Lema de Hintikka – Lema de Hintikka

■ Teorema (Lema de Hintikka)

Todo conjunto de Hintikka es satisfacible.

Demostración en Isabelle:

```
lemma Hintikka-model:  
assumes Hintikka S  
shows isModelSet (setValuation S) S  
proof -  
have  $\forall F. (F \in S \longrightarrow \text{isModel} (\text{setValuation } S) F)$   
proof (rule allI)  
fix F  
have  $(F \in S \longrightarrow \text{isModel} (\text{setValuation } S) F)$   
 $\wedge (\neg F \in S \longrightarrow (\neg(\text{isModel} (\text{setValuation } S) F)))$   
using assms by (rule Hintikkas-lemma-l2)  
thus  $F \in S \longrightarrow \text{isModel} (\text{setValuation } S) F$   
by (rule conjunctI)  
qed  
thus isModelSet (setValuation S) S  
by (simp only: modelSet)  
qed
```



```
theorem  
assumes Hintikka S  
shows sat S  
proof -  
have isModelSet (setValuation S) S  
using assms by (rule Hintikka-model)  
then have  $\exists \mathcal{A}. \text{isModelSet } \mathcal{A} S$   
by (simp only: exI)  
thus sat S  
by (simp only: satAlt)
```

qed

1. La interpretación asociada a un conjunto de Hintikka es modelo del conjunto.

2. Todo conjunto de Hintikka es satisfacible.

1 Objetivo: motivación y contexto del problema

- Desarrollo histórico de la lógica
- Lógica y computación
- Demostradores interactivos: Isabelle

2 Desarrollo

- Métodos en Isabelle/HOL
- Sintaxis
- Semántica
- Lema de Hintikka

3 Conclusiones

Conclusiones sobre el trabajo

- Uso de la plataforma **GitHub**.
- Librerías de Isabelle como herramienta **didáctica**.
- Potencia de **razonamiento automático** del demostrador.
 - **Demostración detallada** (gran extensión, precisa de lemas auxiliares)
 - **Demostración automática** (escasas líneas necesarias)
- Ventajas deductivas del **demostrador interactivo**:
 - 1 **Refuta hipótesis** → *Quickcheck*
 - 2 **Guía en la demostración** → *Sledgehammer*,
find_theorems, *simp_trace*