

Foundations of Artificial Intelligence

Chapter 08: **First-Order Logic**

Roberto Sebastiani

DISI, Università di Trento, Italy – `roberto.sebastiani@unitn.it`

`https://disi.unitn.it/rseba/DIDATTICA/fai\_2026/`

Teaching assistant: **Nicola Debole**, `nicola.debole@unitn.it`,

M.S. Course “Artificial Intelligence Systems”, academic year 2026-2027

Last update: Monday 7th September, 2026, 15:24

Copyright notice: Most examples and images displayed in the slides of this course are taken from [Russell & Norvig, “Artificial Intelligence, a Modern Approach”, 3rd and 4th ed., Pearson], including explicitly figures from the above-mentioned books, so that their copyright is detained by the authors. A few other material (text, figures, examples) is authored by (in alphabetical order): Pieter Abbeel, Bonnie J. Dorr, Anca Dragan, Dan Klein, Nikita Kitaev, Tom Lenaerts, Michela Milano, Dana Nau, Maria Simi, who detain its copyright.

These slides cannot be displayed in public without the permission of the author.

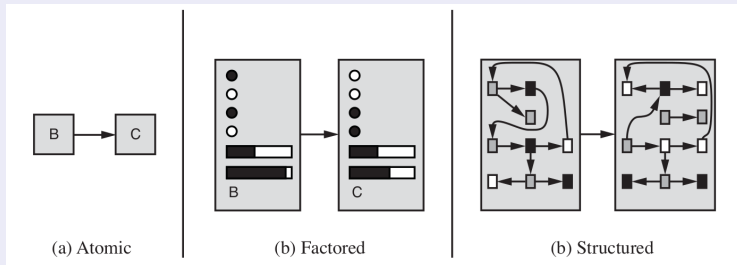
- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Recall: State Representations [Ch. 02]

Representations of states and transitions

- Three ways to represent states and transitions between them:
 - **atomic**: a state is a black box with no internal structure
 - **factored**: a state consists of a vector of attribute values
 - **structured**: a state includes objects, each of which may have attributes of its own as well as relationships to other objects
- increasing expressive power and computational complexity
- reality represented at different levels of abstraction



Pros of Propositional Logic

- PL language **is formal**
 - non-ambiguous semantics
 - unlike natural language, which is intrinsically ambiguous (ex “key”)
- PL **is declarative**
 - knowledge and inference are separate
 - inference is entirely domain independent
- PL **allows for partial/disjunctive/negated information**
 - unlike, e.g., data bases
- PL **is compositional**
 - the meaning of $(A \wedge B) \rightarrow C$ derives from the meaning of A,B,C
- The meaning of PL sentence is **context independent**
 - unlike with natural language, where meaning depends on context

Cons of Propositional Logic

- Is “**Atomic**”: based on **atomic events** which cannot be decomposed
- Assumes the world contains **facts** in the world that are either true or false, nothing else
 - ex: **Man_Socrates**, **Man_Plato**, **Man_Aristotle**, ... distinct atoms

⇒ PL has **has very limited expressive power**

- unlike natural language
- cannot concisely describe an environment with many objects
- e.g., cannot say “**pits cause breezes in adjacent squares**”
(need writing one sentence for each square)

Logics

- A logic is a triple $\langle \mathcal{L}, \mathcal{S}, \mathcal{R} \rangle$ where
 - $\mathcal{L}$, the logic's **language**: a class of sentences described by a formal grammar
 - $\mathcal{S}$, the logic's **semantics**: a formal specification of how to assign meaning in the “real world” to the elements of $\mathcal{L}$
 - $\mathcal{R}$, the logic's **inference system**: is a set of formal derivation rules over $\mathcal{L}$
- There are several logics:
 - propositional logic (PL)
 - first-order logic (FOL)
 - modal logics (MLs)
 - description logics (DLs)
 - temporal logics (TLs)
 - (fuzzy logics, probabilistic logics, ...)
 - ...

- A logic is a triple $\langle \mathcal{L}, \mathcal{S}, \mathcal{R} \rangle$ where
 - $\mathcal{L}$, the logic's **language**: a class of sentences described by a formal grammar
 - $\mathcal{S}$, the logic's **semantics**: a formal specification of how to assign meaning in the “real world” to the elements of $\mathcal{L}$
 - $\mathcal{R}$, the logic's **inference system**: is a set of formal derivation rules over $\mathcal{L}$
- There are several logics:
 - **propositional** logic (PL)
 - **first-order** logic (FOL)
 - **modal** logics (MLs)
 - **description** logics (DLs)
 - **temporal** logics (TLs)
 - (fuzzy logics, probabilistic logics, ...)
 - ...

First-Order Logic (FOL)

- Is **structured**: a world/state includes objects, each of which may have attributes of its own as well as relationships to other objects
- Assumes the world contains:
 - Objects:
e.g., people, houses, numbers, theories, Jim Morrison, colors, basketball games, wars, centuries, ...
 - Relations:
e.g., red, round, bogus, prime, tall ...,
brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
 - Functions:
e.g., father of, best friend, one more than, end of, ...
- Allows to **quantify** on objects
 - ex: “All man are equal”, “some persons are left-handed”, ...

First-Order Logic (FOL)

- Is **structured**: a world/state includes objects, each of which may have attributes of its own as well as relationships to other objects
- Assumes the world contains:
 - **Objects**:
e.g., people, houses, numbers, theories, Jim Morrison, colors, basketball games, wars, centuries, ...
 - **Relations**:
e.g., red, round, bogus, prime, tall ...,
brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
 - **Functions**:
e.g., father of, best friend, one more than, end of, ...
- Allows to **quantify** on objects
 - ex: “All man are equal”, “some persons are left-handed”, ...

First-Order Logic (FOL)

- Is **structured**: a world/state includes objects, each of which may have attributes of its own as well as relationships to other objects
- Assumes the world contains:
 - **Objects**:
e.g., people, houses, numbers, theories, Jim Morrison, colors, basketball games, wars, centuries, ...
 - **Relations**:
e.g., red, round, bogus, prime, tall ...,
brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
 - **Functions**:
e.g., father of, best friend, one more than, end of, ...
- Allows to **quantify** on objects
 - ex: “All man are equal”, “some persons are left-handed”, ...

- 1 Generalities
- 2 **Syntax and Semantics of FOL**
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- Signature: the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “ ”, “ ”, “ ”, “ ”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- Signature: the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- Signature: the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

Syntax of FOL: Basic Elements

- **Constant symbols:** KingJohn, 2, UniversityofTrento,...
- **Predicate symbols:** Man(.), Brother(.,.), ($. > .$), AllDifferent(...),...
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Brother(.,.)) or **infix** (e.g. ($. > .$))
- **Function symbols:** Sqrt, LeftLeg, MotherOf
 - may have different arities (1,2,3,...)
 - may be **prefix** (e.g. Sqrt(.)) or **infix** (e.g. ($. + .$))
- **Variable symbols:** x, y, a, b, ...
- **Propositional Connectives:** $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- **Equality:** “=” (also “ $\neq$ ” s.t. “ $a \neq b$ ” shortcut for “ $\neg(a = b)$ ”)
- **Quantifiers:** “ $\forall$ ” (“forall”), “ $\exists$ ” (“exists”, aka “for some”)
- **Punctuation Symbols:** “,”, “(”, “)”

- Constants symbols are 0-ary function symbols
- Propositions are 0-ary predicates $\implies$ PL subcase of FOL
- **Signature:** the set of predicate, function & constant symbols

FOL: Syntax

- Terms:

- constant or variable or *function*($term_1, \dots, term_n$)
- ex: KingJohn, x, LeftLeg(Richard), ($z \cdot \log(2)$)
- denote objects in the real world (aka domain)

- Atomic sentences (aka atomic formulas):

- $\top, \perp$
- *proposition* or *predicate*($term_1, \dots, term_n$) or $term_1 = term_2$
- ($Length(LeftLeg(Richard)) > Length(LeftLeg(KingJohn))$)
- denote facts

- Non-atomic sentences/formulas:

- $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta, \alpha \oplus \beta,$
 $\forall x.\alpha, \exists x.\alpha$ s.t. x (typically) occurs in α
- Ex: $\forall y.(Italian(y) \rightarrow President(Mattarella, y))$
 $\exists x \forall y. President(x, y) \rightarrow \forall y \exists x. President(x, y)$
 $\forall x.(P(x) \wedge Q(x)) \leftrightarrow ((\forall x.P(x)) \wedge (\forall x.Q(x)))$
 $\forall x.(((x \geq 0) \wedge (x \leq \pi)) \rightarrow (sin(x) \geq 0))$
- denote (complex) facts

FOL: Syntax

- Terms:

- constant or variable or *function*($term_1, \dots, term_n$)
- ex: KingJohn, x, LeftLeg(Richard), ($z \cdot \log(2)$)
- denote objects in the real world (aka domain)

- Atomic sentences (aka atomic formulas):

- $\top, \perp$
- *proposition* or *predicate*($term_1, \dots, term_n$) or $term_1 = term_2$
- ($Length(LeftLeg(Richard)) > Length(LeftLeg(KingJohn))$)
- denote facts

- Non-atomic sentences/formulas:

- $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta, \alpha \oplus \beta,$
 $\forall x.\alpha, \exists x.\alpha$ s.t. x (typically) occurs in α
- Ex: $\forall y.(Italian(y) \rightarrow President(Mattarella, y))$
 $\exists x\forall y.President(x, y) \rightarrow \forall y\exists x.President(x, y)$
 $\forall x.(P(x) \wedge Q(x)) \leftrightarrow ((\forall x.P(x)) \wedge (\forall x.Q(x)))$
 $\forall x.(((x \geq 0) \wedge (x \leq \pi)) \rightarrow (sin(x) \geq 0))$
- denote (complex) facts

FOL: Syntax

- **Terms:**

- **constant** or **variable** or **function**($term_1, \dots, term_n$)
- ex: KingJohn, x, LeftLeg(Richard), ($z \cdot \log(2)$)
- denote **objects** in the real world (aka domain)

- **Atomic sentences** (aka **atomic formulas**):

- $\top, \perp$
- **proposition** or **predicate**($term_1, \dots, term_n$) or $term_1 = term_2$
- ($Length(LeftLeg(Richard)) > Length(LeftLeg(KingJohn))$)
- denote **facts**

- **Non-atomic sentences/formulas:**

- $\neg \alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta, \alpha \oplus \beta,$
 $\forall x. \alpha, \exists x. \alpha$ s.t. x (typically) occurs in α
- Ex: $\forall y. (Italian(y) \rightarrow President(Mattarella, y))$
 $\exists x \forall y. President(x, y) \rightarrow \forall y \exists x. President(x, y)$
 $\forall x. (P(x) \wedge Q(x)) \leftrightarrow ((\forall x. P(x)) \wedge (\forall x. Q(x)))$
 $\forall x. (((x \geq 0) \wedge (x \leq \pi)) \rightarrow (sin(x) \geq 0))$
- denote (complex) **facts**

FOL: Ground and Closed Formulas

- A term/formula is **ground** iff no variable occurs in it (ex: $2 \geq 1$)
- A formula is **closed** iff all variables occurring in it (if any) are quantified (ex: $\forall x \exists y. (x > y)$)

⇒ Ground formulas are closed, but not vice versa.

FOL: Ground and Closed Formulas

- A term/formula is **ground** iff no variable occurs in it (ex: $2 \geq 1$)
- A formula is **closed** iff all variables occurring in it (if any) are quantified (ex: $\forall x \exists y. (x > y)$)

$\Rightarrow$ Ground formulas are closed, but not vice versa.

POLARITY of subformulas

Polarity: the number of nested negations modulo 2.

- **Positive/negative occurrences**

- φ occurs positively in φ ;
- if $\neg\varphi_1$ occurs positively [negatively] in φ ,
then φ_1 occurs negatively [positively] in φ
- if $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ occur positively [negatively] in φ ,
then φ_1 and φ_2 occur positively [negatively] in φ ;
- if $\varphi_1 \rightarrow \varphi_2$ occurs positively [negatively] in φ ,
then φ_1 occurs negatively [positively] in φ and φ_2 occurs positively [negatively] in φ ;
- if $\varphi_1 \leftrightarrow \varphi_2$ or $\varphi_1 \oplus \varphi_2$ occurs in φ ,
then φ_1 and φ_2 occur positively and negatively in φ ;
- if $\forall x.\varphi_1$ or $\exists x.\varphi_1$ occurs positively [negatively] in φ ,
then φ_1 occurs positively [negatively] in φ

POLARITY of subformulas

Polarity: the number of nested negations modulo 2.

- **Positive/negative occurrences**

- φ occurs positively in φ ;
- if $\neg\varphi_1$ occurs positively [negatively] in φ ,
then φ_1 occurs negatively [positively] in φ
- if $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ occur positively [negatively] in φ ,
then φ_1 and φ_2 occur positively [negatively] in φ ;
- if $\varphi_1 \rightarrow \varphi_2$ occurs positively [negatively] in φ ,
then φ_1 occurs negatively [positively] in φ and φ_2 occurs positively [negatively] in φ ;
- if $\varphi_1 \leftrightarrow \varphi_2$ or $\varphi_1 \oplus \varphi_2$ occurs in φ ,
then φ_1 and φ_2 occur positively and negatively in φ ;
- if $\forall x.\varphi_1$ or $\exists x.\varphi_1$ occurs positively [negatively] in φ ,
then φ_1 occurs positively [negatively] in φ

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - **Semantics**
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Truth in FOL: Intuitions

- Sentences are true with respect to a **model**
 - containing a **domain** and an **interpretation**
- The **domain** contains ≥ 1 objects (**domain elements**) and relations and functions over them
- An **interpretation** specifies referents for
 - **variables** $\rightarrow$ objects
 - **constant symbols** $\rightarrow$ objects
 - **predicate symbols** $\rightarrow$ relations
 - **function symbols** $\rightarrow$ functional relations
- An atomic sentence $P(t_1, \dots, t_n)$ is true in an interpretation iff the objects referred to by $t_1, \dots, t_n$ are in the relation referred to by P

Truth in FOL: Intuitions

- Sentences are true with respect to a **model**
 - containing a **domain** and an **interpretation**
- The **domain** contains ≥ 1 objects (**domain elements**) and relations and functions over them
- An **interpretation** specifies referents for
 - **variables** $\rightarrow$ objects
 - **constant symbols** $\rightarrow$ objects
 - **predicate symbols** $\rightarrow$ relations
 - **function symbols** $\rightarrow$ functional relations
- An atomic sentence $P(t_1, \dots, t_n)$ is true in an interpretation iff the objects referred to by $t_1, \dots, t_n$ are in the relation referred to by P

Truth in FOL: Intuitions

- Sentences are true with respect to a **model**
 - containing a **domain** and an **interpretation**
- The **domain** contains ≥ 1 objects (**domain elements**) and relations and functions over them
- An **interpretation** specifies referents for
 - **variables** $\rightarrow$ objects
 - **constant symbols** $\rightarrow$ objects
 - **predicate symbols** $\rightarrow$ relations
 - **function symbols** $\rightarrow$ functional relations
- An atomic sentence $P(t_1, \dots, t_n)$ is true in an interpretation iff the objects referred to by $t_1, \dots, t_n$ are in the relation referred to by P

Truth in FOL: Intuitions

- Sentences are true with respect to a **model**
 - containing a **domain** and an **interpretation**
- The **domain** contains ≥ 1 objects (**domain elements**) and relations and functions over them
- An **interpretation** specifies referents for
 - **variables** $\rightarrow$ objects
 - **constant symbols** $\rightarrow$ objects
 - **predicate symbols** $\rightarrow$ relations
 - **function symbols** $\rightarrow$ functional relations
- An atomic sentence $P(t_1, \dots, t_n)$ is true in an interpretation iff the objects referred to by $t_1, \dots, t_n$ are in the relation referred to by P

FOL: Semantics

FOL Models (aka possible worlds)

- A model $\mathcal{M}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle$ domain, interpretation $\rangle$)
 - Domain $\mathcal{D}$: a non-empty set of objects (aka domain elements)
 - Interpretation $\mathcal{I}$: a (non-injective) map on elements of the signature
 - constant symbols $\mapsto$ domain elements:
a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
 - predicate symbols $\mapsto$ domain relations:
a k -ary predicate $P(\dots)$ is mapped into a subset $[P]^{\mathcal{I}}$ of $\mathcal{D}^k$
(i.e., the set of object tuples satisfying the predicate in this world)
 - functions symbols $\mapsto$ domain functions:
a k -ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \mapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)
- (we denote by $[\cdot]^{\mathcal{I}}$ the result of the interpretation $\mathcal{I}$)

An Interpretation $\mathcal{I}$ is extended to assign domain values to variables, domain values to terms and truth values to formulas.

FOL: Semantics

FOL Models (aka possible worlds)

- A model $\mathcal{M}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle \text{domain}, \text{interpretation} \rangle$)
- Domain $\mathcal{D}$: a **non-empty** set of objects (aka **domain elements**)
- Interpretation $\mathcal{I}$: a (non-injective) map on elements of the signature
 - **constant symbols** $\mapsto$ **domain elements**:
a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
 - **predicate symbols** $\mapsto$ **domain relations**:
a k -ary predicate $P(\dots)$ is mapped into a subset $[P]^{\mathcal{I}}$ of $\mathcal{D}^k$
(i.e., the set of object tuples satisfying the predicate in this world)
 - **functions symbols** $\mapsto$ **domain functions**:
a k -ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \mapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)

(we denote by $[\cdot]^{\mathcal{I}}$ the result of the interpretation $\mathcal{I}$)

An **Interpretation** $\mathcal{I}$ is extended to assign domain values to variables, domain values to terms and truth values to formulas.

FOL: Semantics

FOL Models (aka possible worlds)

- A model $\mathcal{M}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle \text{domain}, \text{interpretation} \rangle$)
 - Domain $\mathcal{D}$: a **non-empty** set of objects (aka **domain elements**)
 - Interpretation $\mathcal{I}$: a (non-injective) map on elements of the signature
 - **constant symbols** $\mapsto$ **domain elements**:
a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
 - **predicate symbols** $\mapsto$ **domain relations**:
a k -ary predicate $P(\dots)$ is mapped into a subset $[P]^{\mathcal{I}}$ of $\mathcal{D}^k$
(i.e., the set of object tuples satisfying the predicate in this world)
 - **functions symbols** $\mapsto$ **domain functions**:
a k -ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \mapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)
- (we denote by $[\cdot]^{\mathcal{I}}$ the result of the interpretation $\mathcal{I}$)

An **Interpretation** $\mathcal{I}$ is extended to assign domain values to variables, domain values to terms and truth values to formulas.

FOL: Semantics

FOL Models (aka possible worlds)

- A model $\mathcal{M}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle \text{domain}, \text{interpretation} \rangle$)
 - Domain $\mathcal{D}$: a **non-empty** set of objects (aka **domain elements**)
 - Interpretation $\mathcal{I}$: a (non-injective) map on elements of the signature
 - **constant symbols** $\mapsto$ **domain elements**:
a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
 - **predicate symbols** $\mapsto$ **domain relations**:
a k -ary predicate $P(\dots)$ is mapped into a subset $[P]^{\mathcal{I}}$ of $\mathcal{D}^k$
(i.e., the set of object tuples satisfying the predicate in this world)
 - **functions symbols** $\mapsto$ **domain functions**:
a k -ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \mapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)
- (we denote by $[\cdot]^{\mathcal{I}}$ the result of the interpretation $\mathcal{I}$)

An **Interpretation** $\mathcal{I}$ is extended to assign domain values to variables, domain values to terms and truth values to formulas.

FOL: Semantics [cont.]

Interpretation of terms

$\mathcal{I}$ maps terms into domain elements

- Variables are assigned domain values
 - **variables** $\mapsto$ **domain elements**:
a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, \dots, t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, \dots, t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, \dots, t_k$:
 - $[f(t_1, \dots, t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}})$
 - Ex: if “Me, Mother, Father” are interpreted as usual, then “Mother(Father(Me))” is interpreted as my (paternal) grandmother
 - Ex: if “+ , − , ⋅ , 0 , 1 , 2 , 3 , 4” are interpreted as usual, then “ $(3 - 1) \cdot (0 + 2)$ ” is interpreted as 4

FOL: Semantics [cont.]

Interpretation of terms

$\mathcal{I}$ maps terms into domain elements

- Variables are assigned domain values
 - **variables** $\mapsto$ **domain elements**:
a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, \dots, t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, \dots, t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, \dots, t_k$:
 - $[f(t_1, \dots, t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}})$
 - Ex: if “Me, Mother, Father” are interpreted as usual, then “Mother(Father(Me))” is interpreted as my (paternal) grandmother
 - Ex: if “+ , − , ⋅ , 0 , 1 , 2 , 3 , 4” are interpreted as usual, then “ $(3 - 1) \cdot (0 + 2)$ ” is interpreted as 4

FOL: Semantics [cont.]

Interpretation of terms

$\mathcal{I}$ maps terms into domain elements

- Variables are assigned domain values
 - **variables** $\mapsto$ **domain elements**:
a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, \dots, t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, \dots, t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, \dots, t_k$:
 - $[f(t_1, \dots, t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}})$
 - Ex: if “**Me**, **Mother**, **Father**” are interpreted as usual, then “**Mother(Father(Me))**” is interpreted as my (paternal) grandmother
 - Ex: if “**+**, **-**, **.**, **0**, **1**, **2**, **3**, **4**” are interpreted as usual, then “ **$(3 - 1) \cdot (0 + 2)$** ” is interpreted as **4**

FOL: Semantics [cont.]

Interpretation of formulas

$\mathcal{I}$ maps formulas into truth values

- An atomic formula $P(t_1, \dots, t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms $t_1, \dots, t_k$ are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
 - $[P(t_1, \dots, t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
 - Ex: if “Me, Mother, Father, Married” are interpreted as tradition, then “Married(Mother(Me), Father(Me))” is interpreted as true
 - Ex: if “+, −, >, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 0) > (1 + 2)$ ” is interpreted as true
- An atomic formula $t_1 = t_2$ is true in $\mathcal{I}$ iff the terms t_1, t_2 are mapped by $\mathcal{I}$ into the same domain element
 - $[t_1 = t_2]^{\mathcal{I}}$ is true iff $[t_1]^{\mathcal{I}}$ same as $[t_2]^{\mathcal{I}}$
 - Ex: if “Mother” is interpreted as usual, Richard, John are brothers, then “Mother(Richard)=Mother(John)” is interpreted as true
 - Ex: if “+, −, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 1) = (1 + 2)$ ” is interpreted as true
- $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$ interpreted by $\mathcal{I}$ as in PL

FOL: Semantics [cont.]

Interpretation of formulas

$\mathcal{I}$ maps formulas into truth values

- An atomic formula $P(t_1, \dots, t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms $t_1, \dots, t_k$ are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
 - $[P(t_1, \dots, t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
 - Ex: if “Me, Mother, Father, Married” are interpreted as tradition, then “Married(Mother(Me), Father(Me))” is interpreted as true
 - Ex: if “+, −, >, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 0) > (1 + 2)$ ” is interpreted as true
- An atomic formula $t_1 = t_2$ is true in $\mathcal{I}$ iff the terms t_1, t_2 are mapped by $\mathcal{I}$ into the same domain element
 - $[t_1 = t_2]^{\mathcal{I}}$ is true iff $[t_1]^{\mathcal{I}}$ same as $[t_2]^{\mathcal{I}}$
 - Ex: if “Mother” is interpreted as usual, Richard, John are brothers, then “Mother(Richard)=Mother(John)” is interpreted as true
 - Ex: if “+, −, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 1) = (1 + 2)$ ” is interpreted as true
- $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$ interpreted by $\mathcal{I}$ as in PL

FOL: Semantics [cont.]

Interpretation of formulas

$\mathcal{I}$ maps formulas into truth values

- An atomic formula $P(t_1, \dots, t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms $t_1, \dots, t_k$ are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
 - $[P(t_1, \dots, t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
 - Ex: if “Me, Mother, Father, Married” are interpreted as tradition, then “Married(Mother(Me), Father(Me))” is interpreted as true
 - Ex: if “+, −, >, 0, 1, 2, 3, 4” are interpreted as usual, then “(4 − 0) > (1 + 2)” is interpreted as true
- An atomic formula $t_1 = t_2$ is true in $\mathcal{I}$ iff the terms t_1, t_2 are mapped by $\mathcal{I}$ into the same domain element
 - $[t_1 = t_2]^{\mathcal{I}}$ is true iff $[t_1]^{\mathcal{I}}$ same as $[t_2]^{\mathcal{I}}$
 - Ex: if “Mother” is interpreted as usual, Richard, John are brothers, then “Mother(Richard)=Mother(John)” is interpreted as true
 - Ex: if “+, −, 0, 1, 2, 3, 4” are interpreted as usual, then “(4 − 1) = (1 + 2)” is interpreted as true
- $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$ interpreted by $\mathcal{I}$ as in PL

FOL: Semantics [cont.]

Interpretation of formulas

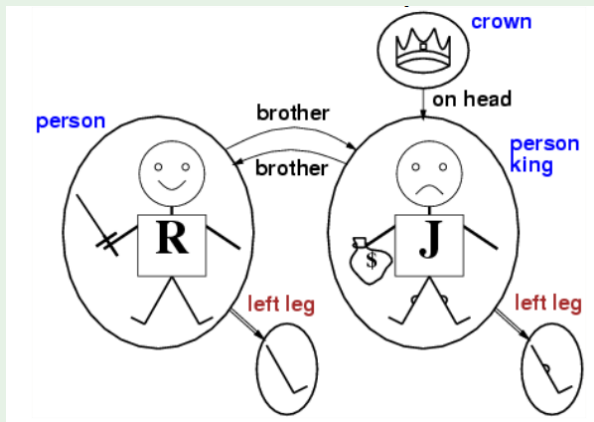
$\mathcal{I}$ maps formulas into truth values

- An atomic formula $P(t_1, \dots, t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms $t_1, \dots, t_k$ are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
 - $[P(t_1, \dots, t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, \dots, [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
 - Ex: if “Me, Mother, Father, Married” are interpreted as tradition, then “Married(Mother(Me), Father(Me))” is interpreted as true
 - Ex: if “+, −, >, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 0) > (1 + 2)$ ” is interpreted as true
- An atomic formula $t_1 = t_2$ is true in $\mathcal{I}$ iff the terms t_1, t_2 are mapped by $\mathcal{I}$ into the same domain element
 - $[t_1 = t_2]^{\mathcal{I}}$ is true iff $[t_1]^{\mathcal{I}}$ same as $[t_2]^{\mathcal{I}}$
 - Ex: if “Mother” is interpreted as usual, Richard, John are brothers, then “Mother(Richard)=Mother(John)” is interpreted as true
 - Ex: if “+, −, 0, 1, 2, 3, 4” are interpreted as usual, then “ $(4 - 1) = (1 + 2)$ ” is interpreted as true
- $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$ interpreted by $\mathcal{I}$ as in PL

Models for FOL: Example

Richard Lionheart and John Lackland

- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
 - $[Richard]^{\mathcal{I}}$: Richard the Lionheart
 - $[John]^{\mathcal{I}}$: evil King John
 - $[Brother]^{\mathcal{I}}$: brotherhood
- $[Brother(Richard, John)]^{\mathcal{I}}$ is true
- $[LeftLeg]^{\mathcal{I}}$ maps any individual to his left leg
- ...

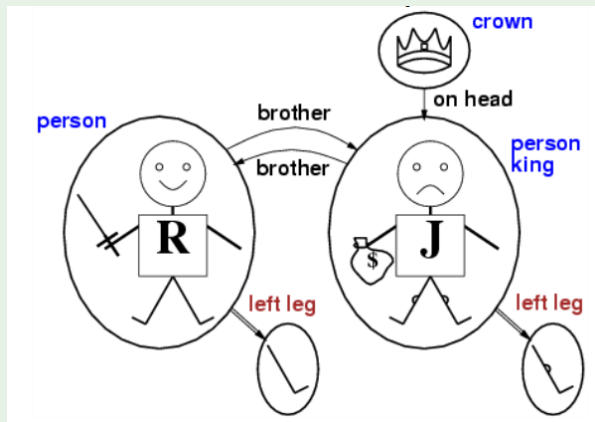


(© S. Russell & P. Norwig, AIMA)

Models for FOL: Example

Richard Lionheart and John Lackland

- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
 - $[\text{Richard}]^{\mathcal{I}}$: Richard the Lionheart
 - $[\text{John}]^{\mathcal{I}}$: evil King John
 - $[\text{Brother}]^{\mathcal{I}}$: brotherhood
- $[\text{Brother}(\text{Richard}, \text{John})]^{\mathcal{I}}$ is true
- $[\text{LeftLeg}]^{\mathcal{I}}$ maps any individual to his left leg
- ...

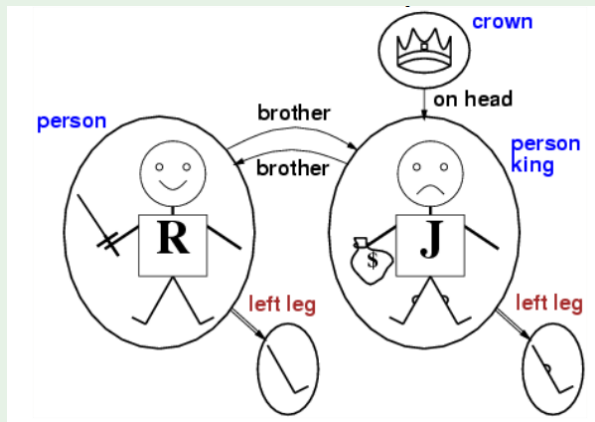


(© S. Russell & P. Norwig, AIMA)

Models for FOL: Example

Richard Lionheart and John Lackland

- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
 - $[\text{Richard}]^{\mathcal{I}}$: Richard the Lionheart
 - $[\text{John}]^{\mathcal{I}}$: evil King John
 - $[\text{Brother}]^{\mathcal{I}}$: brotherhood
- $[\text{Brother}(\text{Richard}, \text{John})]^{\mathcal{I}}$ is true
- $[\text{LeftLeg}]^{\mathcal{I}}$ maps any individual to his left leg
- ...

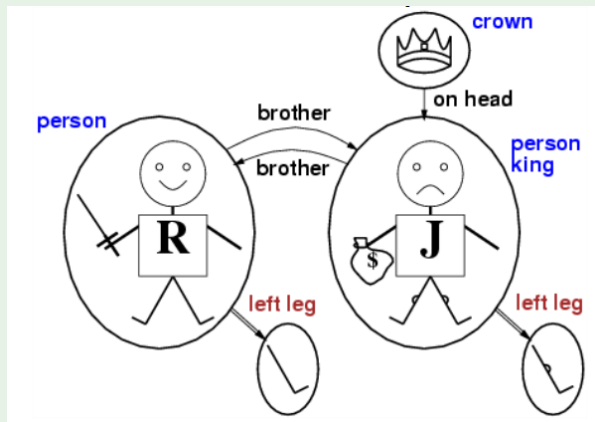


(© S. Russell & P. Norwig, AIMA)

Models for FOL: Example

Richard Lionheart and John Lackland

- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
 - $[Richard]^{\mathcal{I}}$: Richard the Lionheart
 - $[John]^{\mathcal{I}}$: evil King John
 - $[Brother]^{\mathcal{I}}$: brotherhood
- $[Brother(Richard, John)]^{\mathcal{I}}$ is true
- $[LeftLeg]^{\mathcal{I}}$ maps any individual to his left leg
- ...



(© S. Russell & P. Norwig, AIMA)

Models for FOL: Remark

- $[f]^{\mathcal{I}}$ total: must provide an output for every input
- e.g.: $[LeftLeg(crown)]^{\mathcal{I}}$?
- possible solution: assume “null” object ($[LeftLeg(crown) = null]^{\mathcal{I}}$)
(other solution, sorts, not considered here)

Universal Quantification

- $\forall x. \alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\forall x. (\text{King}(x) \rightarrow \text{Person}(x))$ (“all kings are persons”)
- $\forall x. \alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for every possible domain value x is mapped to
- Roughly speaking, can be seen as a conjunction over all (typically infinite) possible instantiations of x in α

$(\text{King}(\text{John}))$	$\rightarrow \text{Person}(\text{John})$	) $\wedge$
$(\text{King}(\text{Richard}))$	$\rightarrow \text{Person}(\text{Richard})$	) $\wedge$
$(\text{King}(\text{crown}))$	$\rightarrow \text{Person}(\text{crown})$	) $\wedge$
$(\text{King}(\text{LeftLeg}(\text{John})))$	$\rightarrow \text{Person}(\text{LeftLeg}(\text{John}))$	) $\wedge$
$(\text{King}(\text{LeftLeg}(\text{LeftLeg}(\text{John}))))$	$\rightarrow \text{Person}(\text{LeftLeg}(\text{LeftLeg}(\text{John})))$	) $\wedge$
...	...	

Universal Quantification

- $\forall x.\alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\forall x.(King(x) \rightarrow Person(x))$ (“all kings are persons”)
- $\forall x.\alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for every possible domain value x is mapped to
- Roughly speaking, can be seen as a conjunction over all (typically infinite) possible instantiations of x in α

$(King(John))$	$\rightarrow Person(John)$	$) \wedge$
$(King(Richard))$	$\rightarrow Person(Richard)$	$) \wedge$
$(King(crown))$	$\rightarrow Person(crown)$	$) \wedge$
$(King(LeftLeg(John)))$	$\rightarrow Person(LeftLeg(John))$	$) \wedge$
$(King(LeftLeg(LeftLeg(John))))$	$\rightarrow Person(LeftLeg(LeftLeg(John)))$	$) \wedge$
...	...	

Universal Quantification

- $\forall x. \alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\forall x. (\text{King}(x) \rightarrow \text{Person}(x))$ (“all kings are persons”)
- $\forall x. \alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for every possible domain value x is mapped to
- Roughly speaking, can be seen as **a conjunction over all (typically infinite) possible instantiations of x in α**

$(\text{King}(\text{John}))$	$\rightarrow \text{Person}(\text{John})$	$) \wedge$
$(\text{King}(\text{Richard}))$	$\rightarrow \text{Person}(\text{Richard})$	$) \wedge$
$(\text{King}(\text{crown}))$	$\rightarrow \text{Person}(\text{crown})$	$) \wedge$
$(\text{King}(\text{LeftLeg}(\text{John})))$	$\rightarrow \text{Person}(\text{LeftLeg}(\text{John}))$	$) \wedge$
$(\text{King}(\text{LeftLeg}(\text{LeftLeg}(\text{John}))))$	$\rightarrow \text{Person}(\text{LeftLeg}(\text{LeftLeg}(\text{John})))$	$) \wedge$
...	...	

Universal Quantification [cont.]

- One may want to restrict the domain of universal quantification to elements of some kind P
 - ex “forall kings ...”, “forall integer numbers...”
- Idea: use an implication, with restrictive predicate as implicant:
 $\forall x.(P(x) \rightarrow \alpha(x, \dots))$
 - ex “ $\forall x.(King(x) \rightarrow \dots)$ ”, “ $\forall x.(Integer(x) \rightarrow \dots)$ ”,
- Beware of typical mistake: do not use “ $\wedge$ ” instead of “ $\rightarrow$ ”
 - ex: “ $\forall x.(King(x) \wedge Person(x))$ ” means “everything/one is a King and is a Person”
 - ex: “ $\forall x.(King(x) \rightarrow Person(x))$ ” means “everything/one who is a King is a Person” (i.e. “every king is a person”)
- “ $\forall$ ” distributes with “ $\wedge$ ”, but not with “ $\vee$ ”
 - $\forall x.(P(x) \wedge Q(x))$ equivalent to $(\forall x.P(x)) \wedge (\forall x.Q(x))$
 - “Everybody is a king and is a person” same as “Everybody is a king and everybody is a person”
 - $\forall x.(P(x) \vee Q(x))$ not equivalent to $(\forall x.P(x)) \vee (\forall x.Q(x))$:
 - “Everybody is a king or is a peasant” much weaker than “Everybody is a king or everybody is a peasant”
 $(\forall x.P(x)) \vee (\forall x.Q(x)) \models \forall x.(P(x) \vee Q(x))$,
 $\forall x.(P(x) \vee Q(x)) \not\models (\forall x.P(x)) \vee (\forall x.Q(x))$

Universal Quantification [cont.]

- One may want to restrict the domain of universal quantification to elements of some kind P
 - ex “forall kings ...”, “forall integer numbers...”
- Idea: use an implication, with restrictive predicate as implicant:
 $\forall x.(P(x) \rightarrow \alpha(x, \dots))$
 - ex “ $\forall x.(King(x) \rightarrow \dots)$ ”, “ $\forall x.(Integer(x) \rightarrow \dots)$ ”,
- Beware of typical mistake: do not use “ $\wedge$ ” instead of “ $\rightarrow$ ”
 - ex: “ $\forall x.(King(x) \wedge Person(x))$ ” means “everything/one is a King and is a Person”
 - ex: “ $\forall x.(King(x) \rightarrow Person(x))$ ” means “everything/one who is a King is a Person” (i.e. “every king is a person”)
- “ $\forall$ ” distributes with “ $\wedge$ ”, but not with “ $\vee$ ”
 - $\forall x.(P(x) \wedge Q(x))$ equivalent to $(\forall x.P(x)) \wedge (\forall x.Q(x))$
 - “Everybody is a king and is a person” same as “Everybody is a king and everybody is a person”
 - $\forall x.(P(x) \vee Q(x))$ not equivalent to $(\forall x.P(x)) \vee (\forall x.Q(x))$:
 - “Everybody is a king or is a peasant” much weaker than “Everybody is a king or everybody is a peasant”
 $(\forall x.P(x)) \vee (\forall x.Q(x)) \models \forall x.(P(x) \vee Q(x))$,
 $\forall x.(P(x) \vee Q(x)) \not\models (\forall x.P(x)) \vee (\forall x.Q(x))$

Universal Quantification [cont.]

- One may want to restrict the domain of universal quantification to elements of some kind P
 - ex “forall kings ...”, “forall integer numbers...”
- Idea: use an implication, with restrictive predicate as implicant:
 $\forall x.(P(x) \rightarrow \alpha(x, \dots))$
 - ex “ $\forall x.(King(x) \rightarrow \dots)$ ”, “ $\forall x.(Integer(x) \rightarrow \dots)$ ”,
- Beware of typical mistake: do not use “ $\wedge$ ” instead of “ $\rightarrow$ ”
 - ex: “ $\forall x.(King(x) \wedge Person(x))$ ” means “everything/one is a King and is a Person”
 - ex: “ $\forall x.(King(x) \rightarrow Person(x))$ ” means “everything/one who is a King is a Person” (i.e. “every king is a person”)
- “ $\forall$ ” distributes with “ $\wedge$ ”, but not with “ $\vee$ ”
 - $\forall x.(P(x) \wedge Q(x))$ equivalent to $(\forall x.P(x)) \wedge (\forall x.Q(x))$
 - “Everybody is a king and is a person” same as “Everybody is a king and everybody is a person”
 - $\forall x.(P(x) \vee Q(x))$ not equivalent to $(\forall x.P(x)) \vee (\forall x.Q(x))$:
 - “Everybody is a king or is a peasant” much weaker than “Everybody is a king or everybody is a peasant”
 $(\forall x.P(x)) \vee (\forall x.Q(x)) \models \forall x.(P(x) \vee Q(x))$,
 $\forall x.(P(x) \vee Q(x)) \not\models (\forall x.P(x)) \vee (\forall x.Q(x))$

Universal Quantification [cont.]

- One may want to restrict the domain of universal quantification to elements of some kind P
 - ex “forall kings ...”, “forall integer numbers...”
- Idea: use an implication, with restrictive predicate as implicant:
 $\forall x.(P(x) \rightarrow \alpha(x, \dots))$
 - ex “ $\forall x.(King(x) \rightarrow \dots)$ ”, “ $\forall x.(Integer(x) \rightarrow \dots)$ ”,
- Beware of typical mistake: do not use “ $\wedge$ ” instead of “ $\rightarrow$ ”
 - ex: “ $\forall x.(King(x) \wedge Person(x))$ ” means “everything/one is a King and is a Person”
 - ex: “ $\forall x.(King(x) \rightarrow Person(x))$ ” means “everything/one who is a King is a Person” (i.e. “every king is a person”)
- “ $\forall$ ” distributes with “ $\wedge$ ”, but not with “ $\vee$ ”
 - $\forall x.(P(x) \wedge Q(x))$ equivalent to $(\forall x.P(x)) \wedge (\forall x.Q(x))$
 - “Everybody is a king and is a person” same as “Everybody is a king and everybody is a person”
 - $\forall x.(P(x) \vee Q(x))$ not equivalent to $(\forall x.P(x)) \vee (\forall x.Q(x))$:
 - “Everybody is a king or is a peasant” much weaker than “Everybody is a king or everybody is a peasant”
 $(\forall x.P(x)) \vee (\forall x.Q(x)) \models \forall x.(P(x) \vee Q(x))$,
 $\forall x.(P(x) \vee Q(x)) \not\models (\forall x.P(x)) \vee (\forall x.Q(x))$

Universal Quantification [cont.]

- One may want to restrict the domain of universal quantification to elements of some kind P
 - ex “forall kings ...”, “forall integer numbers...”
- Idea: use an implication, with restrictive predicate as implicant:
 $\forall x.(P(x) \rightarrow \alpha(x, \dots))$
 - ex “ $\forall x.(King(x) \rightarrow \dots)$ ”, “ $\forall x.(Integer(x) \rightarrow \dots)$ ”,
- Beware of typical mistake: do not use “ $\wedge$ ” instead of “ $\rightarrow$ ”
 - ex: “ $\forall x.(King(x) \wedge Person(x))$ ” means “everything/one is a King and is a Person”
 - ex: “ $\forall x.(King(x) \rightarrow Person(x))$ ” means “everything/one who is a King is a Person” (i.e. “every king is a person”)
- “ $\forall$ ” distributes with “ $\wedge$ ”, but not with “ $\vee$ ”
 - $\forall x.(P(x) \wedge Q(x))$ equivalent to $(\forall x.P(x)) \wedge (\forall x.Q(x))$
 - “Everybody is a king and is a person” same as “Everybody is a king and everybody is a person”
 - $\forall x.(P(x) \vee Q(x))$ not equivalent to $(\forall x.P(x)) \vee (\forall x.Q(x))$:
 - “Everybody is a king or is a peasant” much weaker than “Everybody is a king or everybody is a peasant”
 $(\forall x.P(x)) \vee (\forall x.Q(x)) \models \forall x.(P(x) \vee Q(x))$,
 $\forall x.(P(x) \vee Q(x)) \not\models (\forall x.P(x)) \vee (\forall x.Q(x))$

Existential Quantification

- $\exists x.\alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\exists x.(King(x) \wedge Evil(x))$ (“there is an evil king”)
 - pronounced “exists x s.t. ...” or “for some x ...”
- $\exists x.\alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for some possible domain value x is mapped to
- Roughly speaking, can be seen as a disjunction over all (typically infinite) possible instantiations of x in α

$(King(Richard))$	$\wedge Evil(Richard)$	) $\vee$
$(King(John))$	$\wedge Evil(John)$	) $\vee$
$(King(crown))$	$\wedge Evil(crown)$	) $\vee$
$(King(LeftLeg(John)))$	$\wedge Evil(LeftLeg(John))$	) $\vee$
$(King(LeftLeg(LeftLeg(John))))$	$\wedge Evil(LeftLeg(LeftLeg(John)))$	) $\vee$
...	...	

Existential Quantification

- $\exists x.\alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\exists x.(King(x) \wedge Evil(x))$ (“there is an evil king”)
 - pronounced “exists x s.t. ...” or “for some x ...”
- $\exists x.\alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for some possible domain value x is mapped to
- Roughly speaking, can be seen as a disjunction over all (typically infinite) possible instantiations of x in α

$(King(Richard))$	$\wedge Evil(Richard)$	) $\vee$
$(King(John))$	$\wedge Evil(John)$	) $\vee$
$(King(crown))$	$\wedge Evil(crown)$	) $\vee$
$(King(LeftLeg(John)))$	$\wedge Evil(LeftLeg(John))$	) $\vee$
$(King(LeftLeg(LeftLeg(John))))$	$\wedge Evil(LeftLeg(LeftLeg(John)))$	) $\vee$
...	...	

Existential Quantification

- $\exists x.\alpha(x, \dots)$ (x variable, typically occurs in x)
 - ex: $\exists x.(King(x) \wedge Evil(x))$ (“there is an evil king”)
 - pronounced “exists x s.t. ...” or “for some x ...”
- $\exists x.\alpha(x, \dots)$ true in $\mathcal{M}$ iff
 α is true in $\mathcal{M}$ for some possible domain value x is mapped to
- Roughly speaking, can be seen as a disjunction over all (typically infinite) possible instantiations of x in α

$(King(Richard))$	$\wedge Evil(Richard)$	$)\vee$
$(King(John))$	$\wedge Evil(John)$	$)\vee$
$(King(crown))$	$\wedge Evil(crown)$	$)\vee$
$(King(LeftLeg(John)))$	$\wedge Evil(LeftLeg(John))$	$)\vee$
$(King(LeftLeg(LeftLeg(John))))$	$\wedge Evil(LeftLeg(LeftLeg(John)))$	$)\vee$
...	...	

Existential Quantification [cont.]

- One may want to restrict the domain of existential quantification to elements of some kind P
 - ex “exists a king s.t. ...”, “for some integer numbers...”
- Idea: use a conjunction with restrictive predicate:
 $\exists x.(P(x) \wedge \alpha(x, \dots))$
 - ex “ $\exists x.(King(x) \wedge \dots)$ ”, “ $\exists x.(Integer(x) \wedge \dots)$ ”,
- Beware of typical mistake: do not use “ $\rightarrow$ ” instead of “ $\wedge$ ”
 - ex: “ $\exists x.(King(x) \rightarrow Evil(x))$ ” means “Someone is not a king or is evil”
 - ex: “ $\exists x.(King(x) \wedge Evil(x))$ ” means “Someone is king and is evil”
(i.e., “Some king is evil”)
- “ $\exists$ ” distributes with “ $\vee$ ”, but not with “ $\wedge$ ”
 - $\exists x.(P(x) \vee Q(x))$ equivalent to $(\exists x.P(x)) \vee (\exists x.Q(x))$
 - “Somebody is a king or is a knight” same as
“Somebody is a king or somebody is a knight”
 - $\exists x.(P(x) \wedge Q(x))$ not equivalent to $(\exists x.P(x)) \wedge (\exists x.Q(x))$
 - “Somebody is a king and is evil” much stronger than
“Somebody is a king and somebody is evil”
 $\exists x.(P(x) \wedge Q(x)) \models (\exists x.P(x)) \wedge (\exists x.Q(x))$
 $(\exists x.P(x)) \wedge (\exists x.Q(x)) \not\models \exists x.(P(x) \wedge Q(x))$

Existential Quantification [cont.]

- One may want to restrict the domain of existential quantification to elements of some kind P
 - ex “exists a king s.t. ...”, “for some integer numbers...”
- Idea: **use a conjunction with restrictive predicate:**
 $\exists x.(P(x) \wedge \alpha(x, \dots))$
 - ex “ $\exists x.(King(x) \wedge \dots)$ ”, “ $\exists x.(Integer(x) \wedge \dots)$ ”,
- Beware of typical mistake: **do not use “ $\rightarrow$ ” instead of “ $\wedge$ ”**
- ex: “ $\exists x.(King(x) \rightarrow Evil(x))$ ” means “Someone is not a king or is evil”
 - ex: “ $\exists x.(King(x) \wedge Evil(x))$ ” means “Someone is king and is evil” (i.e., “Some king is evil”)
- “ $\exists$ ” **distributes with “ $\vee$ ”, but not with “ $\wedge$ ”**
- $\exists x.(P(x) \vee Q(x))$ equivalent to $(\exists x.P(x)) \vee (\exists x.Q(x))$
 - “Somebody is a king or is a knight” same as “Somebody is a king or somebody is a knight”
 - $\exists x.(P(x) \wedge Q(x))$ not equivalent to $(\exists x.P(x)) \wedge (\exists x.Q(x))$
 - “Somebody is a king and is evil” much stronger than “Somebody is a king and somebody is evil”

$$\begin{aligned} \exists x.(P(x) \wedge Q(x)) &\models (\exists x.P(x)) \wedge (\exists x.Q(x)) \\ (\exists x.P(x)) \wedge (\exists x.Q(x)) &\not\models \exists x.(P(x) \wedge Q(x)) \end{aligned}$$

Existential Quantification [cont.]

- One may want to restrict the domain of existential quantification to elements of some kind P
 - ex “exists a king s.t. ...”, “for some integer numbers...”
- Idea: **use a conjunction with restrictive predicate:**
 $\exists x.(P(x) \wedge \alpha(x, \dots))$
 - ex “ $\exists x.(King(x) \wedge \dots)$ ”, “ $\exists x.(Integer(x) \wedge \dots)$ ”,
- Beware of typical mistake: **do not use “ $\rightarrow$ ” instead of “ $\wedge$ ”**
- ex: “ $\exists x.(King(x) \rightarrow Evil(x))$ ” means “Someone is not a king or is evil”
 - ex: “ $\exists x.(King(x) \wedge Evil(x))$ ” means “Someone is king and is evil”
(i.e., “Some king is evil”)
- “ $\exists$ ” distributes with “ $\vee$ ”, but not with “ $\wedge$ ”
 - $\exists x.(P(x) \vee Q(x))$ equivalent to $(\exists x.P(x)) \vee (\exists x.Q(x))$
 - “Somebody is a king or is a knight” same as
“Somebody is a king or somebody is a knight”
 - $\exists x.(P(x) \wedge Q(x))$ not equivalent to $(\exists x.P(x)) \wedge (\exists x.Q(x))$
 - “Somebody is a king and is evil” much stronger than
“Somebody is a king and somebody is evil”
 $\exists x.(P(x) \wedge Q(x)) \models (\exists x.P(x)) \wedge (\exists x.Q(x))$
 $(\exists x.P(x)) \wedge (\exists x.Q(x)) \not\models \exists x.(P(x) \wedge Q(x))$

Existential Quantification [cont.]

- One may want to restrict the domain of existential quantification to elements of some kind P
 - ex “exists a king s.t. ...”, “for some integer numbers...”
- Idea: **use a conjunction with restrictive predicate:**
 $\exists x.(P(x) \wedge \alpha(x, \dots))$
 - ex “ $\exists x.(King(x) \wedge \dots)$ ”, “ $\exists x.(Integer(x) \wedge \dots)$ ”,
- Beware of typical mistake: **do not use “ $\rightarrow$ ” instead of “ $\wedge$ ”**
- ex: “ $\exists x.(King(x) \rightarrow Evil(x))$ ” means “Someone is not a king or is evil”
 - ex: “ $\exists x.(King(x) \wedge Evil(x))$ ” means “Someone is king and is evil”
(i.e., “Some king is evil”)
- “ $\exists$ ” **distributes with “ $\vee$ ”, but not with “ $\wedge$ ”**
- $\exists x.(P(x) \vee Q(x))$ equivalent to $(\exists x.P(x)) \vee (\exists x.Q(x))$
 - “Somebody is a king or is a knight” same as
“Somebody is a king or somebody is a knight”
 - $\exists x.(P(x) \wedge Q(x))$ not equivalent to $(\exists x.P(x)) \wedge (\exists x.Q(x))$
 - “Somebody is a king and is evil” much stronger than
“Somebody is a king and somebody is evil”
 $\exists x.(P(x) \wedge Q(x)) \models (\exists x.P(x)) \wedge (\exists x.Q(x))$
 $(\exists x.P(x)) \wedge (\exists x.Q(x)) \not\models \exists x.(P(x) \wedge Q(x))$

Existential Quantification [cont.]

- One may want to restrict the domain of existential quantification to elements of some kind P
 - ex “exists a king s.t. ...”, “for some integer numbers...”
- Idea: **use a conjunction with restrictive predicate:**
 $\exists x.(P(x) \wedge \alpha(x, \dots))$
 - ex “ $\exists x.(King(x) \wedge \dots)$ ”, “ $\exists x.(Integer(x) \wedge \dots)$ ”,
- Beware of typical mistake: **do not use “ $\rightarrow$ ” instead of “ $\wedge$ ”**
- ex: “ $\exists x.(King(x) \rightarrow Evil(x))$ ” means “Someone is not a king or is evil”
 - ex: “ $\exists x.(King(x) \wedge Evil(x))$ ” means “Someone is king and is evil”
(i.e., “Some king is evil”)
- “ $\exists$ ” **distributes with “ $\vee$ ”, but not with “ $\wedge$ ”**
- $\exists x.(P(x) \vee Q(x))$ equivalent to $(\exists x.P(x)) \vee (\exists x.Q(x))$
 - “Somebody is a king or is a knight” same as
“Somebody is a king or somebody is a knight”
 - $\exists x.(P(x) \wedge Q(x))$ not equivalent to $(\exists x.P(x)) \wedge (\exists x.Q(x))$
 - “Somebody is a king and is evil” much stronger than
“Somebody is a king and somebody is evil”
 $\exists x.(P(x) \wedge Q(x)) \models (\exists x.P(x)) \wedge (\exists x.Q(x))$
 $(\exists x.P(x)) \wedge (\exists x.Q(x)) \not\models \exists x.(P(x) \wedge Q(x))$

Examples

- Brothers are siblings
 - $\forall x, y. (Brothers(x, y) \rightarrow Siblings(x, y))$
- "Siblings" is symmetric
 - $\forall x, y. (Siblings(x, y) \leftrightarrow Siblings(y, x))$
- One's mother is one's female parent
 - $\forall x, y. (Mother(x, y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
- A first cousin is a child of a parent's sibling
 - $\forall x_1, x_2. (FirstCousin(x_1, x_2) \leftrightarrow$
 $\exists p_1, p_2. (Siblings(p_1, p_2) \wedge Parent(p_1, x_1) \wedge Parent(p_2, x_2)))$
- Dogs are mammals
 - $\forall x. (Dog(x) \rightarrow Mammal(x))$

Examples

- Brothers are siblings
 - $\forall x, y. (Brothers(x, y) \rightarrow Siblings(x, y))$
- “Siblings” is symmetric
 - $\forall x, y. (Siblings(x, y) \leftrightarrow Siblings(y, x))$
- One’s mother is one’s female parent
 - $\forall x, y. (Mother(x, y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
- A first cousin is a child of a parent’s sibling
 - $\forall x_1, x_2. (FirstCousin(x_1, x_2) \leftrightarrow$
 $\exists p_1, p_2. (Siblings(p_1, p_2) \wedge Parent(p_1, x_1) \wedge Parent(p_2, x_2)))$
- Dogs are mammals
 - $\forall x. (Dog(x) \rightarrow Mammal(x))$

Examples

- Brothers are siblings
 - $\forall x, y. (Brothers(x, y) \rightarrow Siblings(x, y))$
- “Siblings” is symmetric
 - $\forall x, y. (Siblings(x, y) \leftrightarrow Siblings(y, x))$
- One’s mother is one’s female parent
 - $\forall x, y. (Mother(x, y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
- A first cousin is a child of a parent’s sibling
 - $\forall x_1, x_2. (FirstCousin(x_1, x_2) \leftrightarrow$
 $\exists p_1, p_2. (Siblings(p_1, p_2) \wedge Parent(p_1, x_1) \wedge Parent(p_2, x_2)))$
- Dogs are mammals
 - $\forall x. (Dog(x) \rightarrow Mammal(x))$

Examples

- Brothers are siblings
 - $\forall x, y. (Brothers(x, y) \rightarrow Siblings(x, y))$
- “Siblings” is symmetric
 - $\forall x, y. (Siblings(x, y) \leftrightarrow Siblings(y, x))$
- One’s mother is one’s female parent
 - $\forall x, y. (Mother(x, y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
- A first cousin is a child of a parent’s sibling
 - $\forall x_1, x_2. (FirstCousin(x_1, x_2) \leftrightarrow$
 $\exists p_1, p_2. (Siblings(p_1, p_2) \wedge Parent(p_1, x_1) \wedge Parent(p_2, x_2)))$
- Dogs are mammals
 - $\forall x. (Dog(x) \rightarrow Mammal(x))$

Examples

- Brothers are siblings
 - $\forall x, y. (Brothers(x, y) \rightarrow Siblings(x, y))$
- “Siblings” is symmetric
 - $\forall x, y. (Siblings(x, y) \leftrightarrow Siblings(y, x))$
- One’s mother is one’s female parent
 - $\forall x, y. (Mother(x, y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
- A first cousin is a child of a parent’s sibling
 - $\forall x_1, x_2. (FirstCousin(x_1, x_2) \leftrightarrow$
 $\exists p_1, p_2. (Siblings(p_1, p_2) \wedge Parent(p_1, x_1) \wedge Parent(p_2, x_2)))$
- Dogs are mammals
 - $\forall x. (Dog(x) \rightarrow Mammal(x))$

Equality

- Equality is a special predicate: $t_1 = t_2$ is true under a given interpretation if and only if t_1 and t_2 refer to the same object

- Ex: definition of *Sibling* in terms of *Parent*

$$\forall x, y. (Siblings(x, y) \leftrightarrow [\neg(x = y) \wedge \exists p_1, p_2. (\neg(p_1 = p_2) \wedge \\ Parent(p_1, x) \wedge Parent(p_2, x) \wedge Parent(p_1, y) \wedge Parent(p_2, y))])$$

- $1 = 2$ and $x * x = x$ are satisfiable (!)
- $2 = 2$ is valid

Equality

- Equality is a special predicate: $t_1 = t_2$ is true under a given interpretation if and only if t_1 and t_2 refer to the same object
- Ex: definition of *Sibling* in terms of *Parent*
$$\forall x, y. (\text{Siblings}(x, y) \leftrightarrow [\neg(x = y) \wedge \exists p_1, p_2. (\neg(p_1 = p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_2, x) \wedge \text{Parent}(p_1, y) \wedge \text{Parent}(p_2, y))])])$$

- $1 = 2$ and $x * x = x$ are satisfiable (!)
- $2 = 2$ is valid

Equality

- Equality is a special predicate: $t_1 = t_2$ is true under a given interpretation if and only if t_1 and t_2 refer to the same object
- Ex: definition of *Sibling* in terms of *Parent*
$$\forall x, y. (\text{Siblings}(x, y) \leftrightarrow [\neg(x = y) \wedge \exists p_1, p_2. (\neg(p_1 = p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_2, x) \wedge \text{Parent}(p_1, y) \wedge \text{Parent}(p_2, y))])])$$

- $1 = 2$ and $x * x = x$ are satisfiable (!)
- $2 = 2$ is valid

Example

- No one is his/her own sibling

- $\forall x. \neg \text{Siblings}(x, x)$

- Sisters are female, brothers are male

- $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$

- Every married person has a spouse

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Married people have spouses

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Only married people have spouses

- $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$

- People cannot be married to their siblings

- $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling

- $\forall x. \neg \text{Siblings}(x, x)$

- Sisters are female, brothers are male

- $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$

- Every married person has a spouse

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Married people have spouses

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Only married people have spouses

- $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$

- People cannot be married to their siblings

- $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling

- $\forall x. \neg \text{Siblings}(x, x)$

- Sisters are female, brothers are male

- $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$

- Every married person has a spouse

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Married people have spouses

- $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$

- Only married people have spouses

- $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$

- People cannot be married to their siblings

- $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example

- No one is his/her own sibling
 - $\forall x. \neg \text{Siblings}(x, x)$
- Sisters are female, brothers are male
 - $\forall x, y. ((\text{Sisters}(x, y) \rightarrow (\text{Female}(x) \wedge \text{Female}(y))) \wedge (\text{Brothers}(x, y) \rightarrow (\text{Male}(x) \wedge \text{Male}(y))))$
- Every married person has a spouse
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Married people have spouses
 - $\forall x. ((\text{Person}(x) \wedge \text{Married}(x)) \rightarrow \exists y. \text{Spouse}(x, y))$
- Only married people have spouses
 - $\forall x, y. ((\text{Person}(x) \wedge \text{Person}(y) \wedge \text{Spouse}(x, y)) \rightarrow (\text{Married}(x) \wedge \text{Married}(y)))$
- People cannot be married to their siblings
 - $\forall x, y. (\text{Spouse}(x, y) \rightarrow \neg \text{Siblings}(x, y))$

Example (cont.)

- Not everybody has a spouse

- $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
- $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$

- Everybody has a mother

- $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$

- Everybody has a mother and only one

- $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Example (cont.)

- Not everybody has a spouse

- $\neg \forall x. (\text{Person}(x) \rightarrow \exists y. \text{Spouse}(x, y))$ or
- $\exists x. (\text{Person}(x) \wedge \neg \exists y. \text{Spouse}(x, y))$

- Everybody has a mother

- $\forall x. (\text{Person}(x) \rightarrow \exists y. \text{Mother}(y, x))$

- Everybody has a mother and only one

- $\forall x. \text{Person}(x) \rightarrow (\exists y. \text{Mother}(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge \text{Mother}(z, x)))$

Example (cont.)

- Not everybody has a spouse

- $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
- $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$

- Everybody has a mother

- $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$

- Everybody has a mother and only one

- $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Example (cont.)

- Not everybody has a spouse

- $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
- $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$

- Everybody has a mother

- $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$

- Everybody has a mother and only one

- $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Example (cont.)

- Not everybody has a spouse

- $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
- $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$

- Everybody has a mother

- $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$

- Everybody has a mother and only one

- $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Example (cont.)

- Not everybody has a spouse
 - $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
 - $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$
- Everybody has a mother
 - $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$
- Everybody has a mother and only one
 - $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Example (cont.)

- Not everybody has a spouse
 - $\neg \forall x. (Person(x) \rightarrow \exists y. Spouse(x, y))$ or
 - $\exists x. (Person(x) \wedge \neg \exists y. Spouse(x, y))$
- Everybody has a mother
 - $\forall x. (Person(x) \rightarrow \exists y. Mother(y, x))$
- Everybody has a mother and only one
 - $\forall x. Person(x) \rightarrow (\exists y. Mother(y, x) \wedge \neg \exists z. (\neg(y = z) \wedge Mother(z, x)))$

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$
(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists y.P(x, y)$ is not the same as $\forall y.\exists y.P(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists y.P(x, y)$ is not the same as $\forall y.\exists y.P(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists y.P(x, y)$ is not the same as $\forall y.\exists y.P(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists y.P(x, y)$ is not the same as $\forall y.\exists y.P(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists y.P(x, y)$ is not the same as $\forall y.\exists y.P(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists yP(x, y)$ is **not** the same as $\forall y.\exists yP(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \iff \forall x\forall y.\alpha \iff \forall x, y.\alpha \iff \forall xy.\alpha$

(same with $\exists$)

- if x does not occur in φ , $\forall x.\varphi$ equivalent to $\exists x.\varphi$ equivalent to φ
- $\forall xy.P(x, y)$ equivalent to $\forall yx.P(x, y)$
 - ex: $\forall xy.(x < y)$ same as $\forall yx.(x < y)$
- $\exists xy.P(x, y)$ equivalent to $\exists yx.P(x, y)$
 - ex: $\exists xy.Twins(x, y)$ same as $\exists yx.Twins(x, y)$
- $\exists x\forall y.P(x, y)$ not equivalent to $\forall y\exists x.P(x, y)$
 - ex: $\forall y\exists x.Father(x, y)$ much weaker than $\exists x\forall y.Father(x, y)$
“everybody has a father” vs. “exists a father of everybody”
 $\exists x\forall y.P(x, y) \models \forall y\exists x.P(x, y)$
 $\forall y\exists x.P(x, y) \not\models \exists x\forall y.P(x, y)$

Remark

- Variable names are irrelevant: e.g., $\forall x.P(x)$ is the same as $\forall y.P(y)$
- ... provided there are no name conflicts: e.g., $\forall x.\exists yP(x, y)$ is **not** the same as $\forall y.\exists yP(y, y)$!

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual

- $\forall x.\alpha \iff \neg \exists x.\neg \alpha$
- $\neg \forall x.\alpha \iff \exists x.\neg \alpha$
- $\exists x.\alpha \iff \neg \forall x.\neg \alpha$
- $\neg \exists x.\alpha \iff \forall x.\neg \alpha$

- Examples

- $\forall x.Likes(x, Icecream)$ equivalent to $\neg \exists x.\neg Likes(x, Icecream)$
- $\exists x.Likes(x, Broccoli)$ equivalent to $\neg \forall x.\neg Likes(x, Broccoli)$

- Negated restricted quantifiers switch “ $\rightarrow$ ” with “ $\wedge$ ”

- $\forall x.(P(x) \rightarrow \alpha) \iff \neg \exists x.(P(x) \wedge \neg \alpha)$
- $\neg \forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg \alpha)$
- ...

- Ex: “not all kings are evil” same as “some king is not evil”

- $\neg \forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$

- Unsurprising, since $\langle \forall, \exists \rangle$ are $\langle \wedge, \vee \rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual

- $\forall x.\alpha \iff \neg \exists x.\neg \alpha$
- $\neg \forall x.\alpha \iff \exists x.\neg \alpha$
- $\exists x.\alpha \iff \neg \forall x.\neg \alpha$
- $\neg \exists x.\alpha \iff \forall x.\neg \alpha$

- Examples

- $\forall x.Likes(x, Icecream)$ equivalent to $\neg \exists x.\neg Likes(x, Icecream)$
- $\exists x.Likes(x, Broccoli)$ equivalent to $\neg \forall x.\neg Likes(x, Broccoli)$

- Negated restricted quantifiers switch “ $\rightarrow$ ” with “ $\wedge$ ”

- $\forall x.(P(x) \rightarrow \alpha) \iff \neg \exists x.(P(x) \wedge \neg \alpha)$
- $\neg \forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg \alpha)$
- ...

- Ex: “not all kings are evil” same as “some king is not evil”

- $\neg \forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$

- Unsurprising, since $\langle \forall, \exists \rangle$ are $\langle \wedge, \vee \rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual

- $\forall x.\alpha \iff \neg \exists x.\neg \alpha$
- $\neg \forall x.\alpha \iff \exists x.\neg \alpha$
- $\exists x.\alpha \iff \neg \forall x.\neg \alpha$
- $\neg \exists x.\alpha \iff \forall x.\neg \alpha$

- Examples

- $\forall x.Likes(x, Icecream)$ equivalent to $\neg \exists x.\neg Likes(x, Icecream)$
- $\exists x.Likes(x, Broccoli)$ equivalent to $\neg \forall x.\neg Likes(x, Broccoli)$

- Negated restricted quantifiers switch “ $\rightarrow$ ” with “ $\wedge$ ”

- $\forall x.(P(x) \rightarrow \alpha) \iff \neg \exists x.(P(x) \wedge \neg \alpha)$
- $\neg \forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg \alpha)$
- ...

- Ex: “not all kings are evil” same as “some king is not evil”

- $\neg \forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$

- Unsurprising, since $\langle \forall, \exists \rangle$ are $\langle \wedge, \vee \rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual

- $\forall x.\alpha \iff \neg \exists x.\neg \alpha$
- $\neg \forall x.\alpha \iff \exists x.\neg \alpha$
- $\exists x.\alpha \iff \neg \forall x.\neg \alpha$
- $\neg \exists x.\alpha \iff \forall x.\neg \alpha$

- Examples

- $\forall x.Likes(x, Icecream)$ equivalent to $\neg \exists x.\neg Likes(x, Icecream)$
- $\exists x.Likes(x, Broccoli)$ equivalent to $\neg \forall x.\neg Likes(x, Broccoli)$

- Negated restricted quantifiers switch “ $\rightarrow$ ” with “ $\wedge$ ”

- $\forall x.(P(x) \rightarrow \alpha) \iff \neg \exists x.(P(x) \wedge \neg \alpha)$
- $\neg \forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg \alpha)$
- ...

- Ex: “not all kings are evil” same as “some king is not evil”

- $\neg \forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$

- Unsurprising, since $\langle \forall, \exists \rangle$ are $\langle \wedge, \vee \rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual

- $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
- $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
- $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
- $\neg\exists x.\alpha \iff \forall x.\neg\alpha$

- Examples

- $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
- $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$

- Negated restricted quantifiers switch “ $\rightarrow$ ” with “ $\wedge$ ”

- $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
- $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
- ...

- Ex: “not all kings are evil” same as “some king is not evil”

- $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$

- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α **entails** β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ **satisfiable** iff $\bigwedge_{i=1}^n \varphi_i$ **satisfiable**
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ **valid** iff $\bigwedge_{i=1}^n \varphi_i$ **valid**

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α entails β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ satisfiable iff $\bigwedge_{i=1}^n \varphi_i$ satisfiable
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ valid iff $\bigwedge_{i=1}^n \varphi_i$ valid

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α **entails** β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ **satisfiable** iff $\bigwedge_{i=1}^n \varphi_i$ **satisfiable**
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ **valid** iff $\bigwedge_{i=1}^n \varphi_i$ **valid**

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α **entails** β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ satisfiable iff $\bigwedge_{i=1}^n \varphi_i$ satisfiable
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ valid iff $\bigwedge_{i=1}^n \varphi_i$ valid

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α **entails** β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ satisfiable iff $\bigwedge_{i=1}^n \varphi_i$ satisfiable
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ valid iff $\bigwedge_{i=1}^n \varphi_i$ valid

Satisfiability, Validity, Entailment

- A model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathcal{D}, \mathcal{I} \rangle$ satisfies φ ($\mathcal{M} \models \varphi$) iff $[\varphi]^{\mathcal{I}}$ is true
- $M(\varphi) \stackrel{\text{def}}{=} \{\mathcal{M} \mid \mathcal{M} \models \varphi\}$ (the set of models of φ)
- φ is **satisfiable** iff $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ (i.e. $M(\varphi) \neq \emptyset$)
- α **entails** β ($\alpha \models \beta$) iff, for all $\mathcal{M}$, $\mathcal{M} \models \alpha \implies \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is **valid** ($\models \varphi$) iff $\mathcal{M} \models \varphi$ for all $\mathcal{M}$ s (i.e., $\mathcal{M} \in M(\varphi)$ for all $\mathcal{M}$ s)
- α, β are **equivalent** iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e. $M(\alpha) = M(\beta)$)

Sets of formulas as conjunctions

Let $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, \dots, \varphi_n\}$. Then:

- Γ satisfiable iff $\bigwedge_{i=1}^n \varphi_i$ satisfiable
- $\Gamma \models \phi$ iff $\bigwedge_{i=1}^n \varphi_i \models \phi$
- Γ valid iff $\bigwedge_{i=1}^n \varphi_i$ valid

Properties & Results

Property

φ is valid iff $\neg\varphi$ is unsatisfiable

Deduction Theorem

$\alpha \models \beta$ iff $\alpha \rightarrow \beta$ is valid ($\models \alpha \rightarrow \beta$)

Corollary

$\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable

Validity and entailment checking can be straightforwardly reduced to (un)satisfiability checking!

Properties & Results

Property

φ is valid iff $\neg\varphi$ is unsatisfiable

Deduction Theorem

$\alpha \models \beta$ iff $\alpha \rightarrow \beta$ is valid ($\models \alpha \rightarrow \beta$)

Corollary

$\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable

Validity and entailment checking can be straightforwardly reduced to (un)satisfiability checking!

Properties & Results

Property

φ is valid iff $\neg\varphi$ is unsatisfiable

Deduction Theorem

$\alpha \models \beta$ iff $\alpha \rightarrow \beta$ is valid ($\models \alpha \rightarrow \beta$)

Corollary

$\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable

Validity and entailment checking can be straightforwardly reduced to (un)satisfiability checking!

Properties & Results

Property

φ is valid iff $\neg\varphi$ is unsatisfiable

Deduction Theorem

$\alpha \models \beta$ iff $\alpha \rightarrow \beta$ is valid ($\models \alpha \rightarrow \beta$)

Corollary

$\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable

Validity and entailment checking can be straightforwardly reduced to (un)satisfiability checking!

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x. P(x) \rightarrow \exists x. P(x)$ valid
- $\forall x. P(x) \models \exists x. P(x)$
- $\neg(\forall x. P(x)) \rightarrow \exists x. P(x)$ unsatisfiable
- $\forall x. P(x) \wedge \neg \exists x. P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Examples

- $P(x), \forall x.(x \geq y), \{\forall x.(x \geq 0), \forall x.(x + 1 > x)\}$ satisfiable (!)
- $P(x) \wedge \neg P(x), \neg(x = x), (\forall x, y. Q(x, y)) \rightarrow \neg Q(a, b)$ unsatisfiable
- $\forall x.P(x) \rightarrow \exists x.P(x)$ valid
- $\forall x.P(x) \models \exists x.P(x)$
- $\neg(\forall x.P(x)) \rightarrow \exists x.P(x)$ unsatisfiable
- $\forall x.P(x) \wedge \neg \exists x.P(x)$ unsatisfiable

$(1 > 2)$ is satisfiable. Why?

Exercises

- Is $\forall x.P(x)$ equivalent to $\forall y.P(y)$?
- Is $\forall xy.P(x, y)$ equivalent to $\forall yx.P(y, x)$?
- $\forall x.\exists x.P(x)$ is equivalent to:
 - $\exists x.P(x)$
 - $\forall x.P(x)$
 - neither
- $\exists x.\forall x.P(x)$ is equivalent to:
 - $\exists x.P(x)$
 - $\forall x.P(x)$
 - neither

Enumeration of Models?

- We *can enumerate* the models for a given FOL sentence:

- For each number of universe elements n from 1 to ∞

- For each k -ary predicate P_k in the sentence

- For each possible k -ary relation on n objects

- For each constant symbol C in the sentence

- For each one of n objects C is mapped to

- ...

- $\implies$ Enumerating models is not going to be easy!

Enumeration of Models?

- We *can enumerate* the models for a given FOL sentence:

- For each number of universe elements n from 1 to ∞

- For each k -ary predicate P_k in the sentence

- For each possible k -ary relation on n objects

- For each constant symbol C in the sentence

- For each one of n objects C is mapped to

- ...

- $\implies$ Enumerating models is not going to be easy!

Semi-decidability of FOL

Theorem

Entailment (validity, unsatisfiability) in FOL is only **semi-decidable**:

- if $\Gamma \models \alpha$, this can be checked in finite time
- if $\Gamma \not\models \alpha$, no algorithm is guaranteed to check it in finite time

Semi-decidability of FOL

Theorem

Entailment (validity, unsatisfiability) in FOL is only **semi-decidable**:

- if $\Gamma \models \alpha$, this can be checked in finite time
- if $\Gamma \not\models \alpha$, no algorithm is guaranteed to check it in finite time



- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Outline

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - **FOL Agents**
 - Example: The Wumpus World

FOL Knowledge-Based Agent

- We can assert FOL sentences (**assertions**) into the KB: **Tell[...]**

- ex: **Tell[KB, King(John)]**
- ex: **Tell[KB, Person(Richard)]**
- ex: **Tell[KB, $\forall x.(King(x) \rightarrow Person(x))$]**

- We can ask **queries** (aka **goals**) to the KB: **Ask[...]**

- ex: **Ask[KB, King(John)]**
- ex: **Ask[KB, Person(John)]**
- ex: **Ask[KB, $\exists x.Person(x)$]**

$\Rightarrow$ **Ask[KB, α]** returns true only if $KB \models \alpha$

- Other queries: **AskVars[...]**, asking for variable values

$\Rightarrow$ returns one (or more) **binding lists** (aka **substitutions**) $\{var/term; var/term, \dots\}$

- ex: **AskVars[KB, $\exists x.Person(x)$]** $\Rightarrow \{x/John\}; \{x/Richard\}$
- typical for Horn clauses
(e.g. with $King(John) \vee King(Richard)$,
the query **AskVars[KB, $\exists x.King(x)$]** would not cause a binding list)

FOL Knowledge-Based Agent

- We can assert FOL sentences (**assertions**) into the KB: **Tell[...]**
 - ex: **Tell[KB, King(John)]**
 - ex: **Tell[KB, Person(Richard)]**
 - ex: **Tell[KB, $\forall x.(King(x) \rightarrow Person(x))$]**
- We can ask **queries** (aka **goals**) to the KB: **Ask[...]**
 - ex: **Ask[KB, King(John)]**
 - ex: **Ask[KB, Person(John)]**
 - ex: **Ask[KB, $\exists x.Person(x)$]**

$\Rightarrow$ **Ask[KB, α]** returns true only if $KB \models \alpha$

- Other queries: **AskVars[...]**, asking for variable values
 $\Rightarrow$ returns one (or more) **binding lists** (aka **substitutions**) $\{var/term; var/term, \dots\}$
 - ex: **AskVars[KB, $\exists x.Person(x)$]** $\Rightarrow \{x/John\}; \{x/Richard\}$
 - typical for Horn clauses
(e.g. with $King(John) \vee King(Richard)$,
the query **AskVars[KB, $\exists x.King(x)$]** would not cause a binding list)

FOL Knowledge-Based Agent

- We can assert FOL sentences (**assertions**) into the KB: **Tell[...]**
 - ex: **Tell[KB, King(John)]**
 - ex: **Tell[KB, Person(Richard)]**
 - ex: **Tell[KB, $\forall x.(King(x) \rightarrow Person(x))$]**
- We can ask **queries** (aka **goals**) to the KB: **Ask[...]**
 - ex: **Ask[KB, King(John)]**
 - ex: **Ask[KB, Person(John)]**
 - ex: **Ask[KB, $\exists x.Person(x)$]**

$\Rightarrow$ **Ask[KB, α]** returns true only if $KB \models \alpha$

- Other queries: **AskVars[...]**, asking for variable values
 $\Rightarrow$ returns one (or more) **binding lists** (aka **substitutions**) $\{var/term; var/term, \dots\}$
 - ex: **AskVars[KB, $\exists x.Person(x)$]** $\Rightarrow \{x/John\}; \{x/Richard\}$
 - typical for Horn clauses
(e.g. with $King(John) \vee King(Richard)$,
the query **AskVars[KB, $\exists x.King(x)$]** would not cause a binding list)

FOL Knowledge-Based Agent

- We can assert FOL sentences (**assertions**) into the KB: **Tell[...]**

- ex: **Tell[KB, King(John)]**
- ex: **Tell[KB, Person(Richard)]**
- ex: **Tell[KB, $\forall x.(King(x) \rightarrow Person(x))$]**

- We can ask **queries** (aka **goals**) to the KB: **Ask[...]**

- ex: **Ask[KB, King(John)]**
- ex: **Ask[KB, Person(John)]**
- ex: **Ask[KB, $\exists x.Person(x)$]**

$\Rightarrow$ **Ask[KB, α]** returns true only if $KB \models \alpha$

- Other queries: **AskVars[...]**, asking for variable values

$\Rightarrow$ returns one (or more) **binding lists** (aka **substitutions**) $\{var/term; var/term, \dots\}$

- ex: **AskVars[KB, $\exists x.Person(x)$]** $\Rightarrow \{x/John\}; \{x/Richard\}$
- typical for Horn clauses
(e.g. with **King(John) $\vee$ King(Richard)**,
the query **AskVars[KB, $\exists x.King(x)$]** would not cause a binding list)

Example: The Kinship Domain

Domain of family relationships

- Binary predicate symbols (family relationships):
 - Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle
- function symbols:
 - Mother, Father
- Knowledge base KB:
 - 1 $\forall x, y. (x = \text{Mother}(y) \leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)))$
 - 2 $\forall x, y. (\text{Brother}(x, y) \leftrightarrow (\text{Male}(x) \wedge \text{Sibling}(x, y)))$
 - 3 $\forall x, y. (\text{Grandparent}(x, y) \leftrightarrow \exists z. (\text{Parent}(x, z) \wedge \text{Parent}(z, y)))$
 - 4 $\forall x, y. (\text{Sibling}(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2. ((p_1 \neq p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_1, y) \wedge (\text{Parent}(p_2, x) \wedge \text{Parent}(p_2, y))))$
 - 5 ...
- Queries inferred from KB
 - ex: $(4) \models \forall x, y. (\text{Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x))$

Notation: “ $t \neq s$ ” shortcut for “ $\neg(t = s)$ ”

Example: The Kinship Domain

Domain of family relationships

- Binary predicate symbols (family relationships):
 - Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle
- function symbols:
 - Mother, Father
- Knowledge base KB:
 - 1 $\forall x, y. (x = \text{Mother}(y) \leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)))$
 - 2 $\forall x, y. (\text{Brother}(x, y) \leftrightarrow (\text{Male}(x) \wedge \text{Sibling}(x, y)))$
 - 3 $\forall x, y. (\text{Grandparent}(x, y) \leftrightarrow \exists z. (\text{Parent}(x, z) \wedge \text{Parent}(z, y)))$
 - 4 $\forall x, y. (\text{Sibling}(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2. ((p_1 \neq p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_1, y) \wedge (\text{Parent}(p_2, x) \wedge \text{Parent}(p_2, y))))$
 - 5 ...
- Queries inferred from KB
 - ex: (4) $\models \forall x, y. (\text{Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x))$

Notation: “ $t \neq s$ ” shortcut for “ $\neg(t = s)$ ”

Example: The Kinship Domain

Domain of family relationships

- Binary predicate symbols (family relationships):
 - Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle
- function symbols:
 - Mother, Father
- Knowledge base KB:
 - 1 $\forall x, y. (x = \text{Mother}(y) \leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)))$
 - 2 $\forall x, y. (\text{Brother}(x, y) \leftrightarrow (\text{Male}(x) \wedge \text{Sibling}(x, y)))$
 - 3 $\forall x, y. (\text{Grandparent}(x, y) \leftrightarrow \exists z. (\text{Parent}(x, z) \wedge \text{Parent}(z, y)))$
 - 4 $\forall x, y. (\text{Sibling}(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2. ((p_1 \neq p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_1, y) \wedge (\text{Parent}(p_2, x) \wedge \text{Parent}(p_2, y))))$
 - 5 ...
- Queries inferred from KB
 - ex: $(4) \models \forall x, y. (\text{Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x))$

Notation: “ $t \neq s$ ” shortcut for “ $\neg(t = s)$ ”

Example: The Kinship Domain

Domain of family relationships

- Binary predicate symbols (family relationships):
 - Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle
- function symbols:
 - Mother, Father
- Knowledge base KB:
 - 1 $\forall x, y. (x = \text{Mother}(y) \leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)))$
 - 2 $\forall x, y. (\text{Brother}(x, y) \leftrightarrow (\text{Male}(x) \wedge \text{Sibling}(x, y)))$
 - 3 $\forall x, y. (\text{Grandparent}(x, y) \leftrightarrow \exists z. (\text{Parent}(x, z) \wedge \text{Parent}(z, y)))$
 - 4 $\forall x, y. (\text{Sibling}(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2. ((p_1 \neq p_2) \wedge \text{Parent}(p_1, x) \wedge \text{Parent}(p_1, y) \wedge (\text{Parent}(p_2, x) \wedge \text{Parent}(p_2, y))))$
 - 5 ...
- Queries inferred from KB
 - ex: (4) $\models \forall x, y. (\text{Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x))$

Notation: “ $t \neq s$ ” shortcut for “ $\neg(t = s)$ ”

Example: Integer Numbers

Peano Arithmetic

- Basic symbols

- Unary predicate symbol: **NatNum** (natural number)
- Unary function symbol: **S** (Successor)
- Constant symbol: **0**

- Defined symbols:

- Binary function symbols: $+$, $*$ (infix)
- Constant symbols: $1, 2, 3, 4, 5, 6, \dots$

- Knowledge base KB:

- 1 $\text{NatNum}(0)$
- 2 $\forall x. (\text{NatNum}(x) \rightarrow \text{NatNum}(S(x)))$
- 3 $\forall x. (\text{NatNum}(x) \rightarrow (0 \neq S(x)))$
- 4 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
- 5 $\forall x. (\text{NatNum}(x) \rightarrow (x = (0 + x)))$
- 6 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow (S(x) + y = S(x + y)))$
- 7 $1 = S(0), 2 = S(1), 3 = S(2), \dots$

- Queries inferred from KB

- ex: $(4) \models \forall x, y. ((\text{NatNum}(x) \wedge (\text{NatNum}(y))) \rightarrow ((x + y) = (y + x)))$

Example: Integer Numbers

Peano Arithmetic

- Basic symbols
 - Unary predicate symbol: NatNum (natural number)
 - Unary function symbol: S (Successor)
 - Constant symbol: 0
- Defined symbols:
 - Binary function symbols: $+, *$ (infix)
 - Constant symbols: $1, 2, 3, 4, 5, 6, \dots$
- Knowledge base KB:
 - 1 $\text{NatNum}(0)$
 - 2 $\forall x. (\text{NatNum}(x) \rightarrow \text{NatNum}(S(x)))$
 - 3 $\forall x. (\text{NatNum}(x) \rightarrow (0 \neq S(x)))$
 - 4 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
 - 5 $\forall x. (\text{NatNum}(x) \rightarrow (x = (0 + x)))$
 - 6 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow (S(x) + y = S(x + y)))$
 - 7 $1 = S(0), 2 = S(1), 3 = S(2), \dots$
- Queries inferred from KB
 - ex: $(4) \models \forall x, y. ((\text{NatNum}(x) \wedge (\text{NatNum}(y))) \rightarrow ((x + y) = (y + x)))$

Example: Integer Numbers

Peano Arithmetic

- Basic symbols
 - Unary predicate symbol: NatNum (natural number)
 - Unary function symbol: S (Successor)
 - Constant symbol: 0
- Defined symbols:
 - Binary function symbols: $+, *$ (infix)
 - Constant symbols: $1, 2, 3, 4, 5, 6, \dots$
- Knowledge base KB:
 - 1 $\text{NatNum}(0)$
 - 2 $\forall x. (\text{NatNum}(x) \rightarrow \text{NatNum}(S(x)))$
 - 3 $\forall x. (\text{NatNum}(x) \rightarrow (0 \neq S(x)))$
 - 4 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
 - 5 $\forall x. (\text{NatNum}(x) \rightarrow (x = (0 + x)))$
 - 6 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow (S(x) + y = S(x + y)))$
 - 7 $1 = S(0), 2 = S(1), 3 = S(2), \dots$
- Queries inferred from KB
 - ex: $(4) \models \forall x, y. ((\text{NatNum}(x) \wedge (\text{NatNum}(y))) \rightarrow ((x + y) = (y + x)))$

Example: Integer Numbers

Peano Arithmetic

- Basic symbols
 - Unary predicate symbol: NatNum (natural number)
 - Unary function symbol: S (Successor)
 - Constant symbol: 0
- Defined symbols:
 - Binary function symbols: $+, *$ (infix)
 - Constant symbols: $1, 2, 3, 4, 5, 6, \dots$
- Knowledge base KB:
 - 1 $\text{NatNum}(0)$
 - 2 $\forall x. (\text{NatNum}(x) \rightarrow \text{NatNum}(S(x)))$
 - 3 $\forall x. (\text{NatNum}(x) \rightarrow (0 \neq S(x)))$
 - 4 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
 - 5 $\forall x. (\text{NatNum}(x) \rightarrow (x = (0 + x)))$
 - 6 $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow (S(x) + y = S(x + y)))$
 - 7 $1 = S(0), 2 = S(1), 3 = S(2), \dots$
- Queries inferred from KB
 - ex: $(4) \models \forall x, y. ((\text{NatNum}(x) \wedge (\text{NatNum}(y))) \rightarrow ((x + y) = (y + x)))$

Exercises

About the Kinship domain

- Try to add the axioms defining other predicates or functions (e.g. Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle, ...)
- Add some ground atom or its negation to the KB (ex: Brother(Steve,Mary), Mary=Mother(Paul),...)
- Try to solve some query by entailment (e.g. Uncle(Steve,Paul), $\exists x. \text{Uncle}(x, \text{Paul})$, ...)

About the Peano Arithmetic domain

- Try to add the axioms defining other predicate or functions (e.g. " $n \leq m$ " or " $m * n$ ", " n^m ")
- Add some ground atom or its negation to the KB (ex: $1 = S(0)$, $2 = S(1)$, ...)
- Try to solve some query by entailment (e.g. $3 + 2 = 5$, $2 * 3 = 6$, ...)

Exercises

About the Kinship domain

- Try to add the axioms defining other predicates or functions (e.g. Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle, ...)
- Add some ground atom or its negation to the KB (ex: Brother(Steve,Mary), Mary=Mother(Paul),...)
- Try to solve some query by entailment (e.g. Uncle(Steve,Paul), $\exists x. \text{Uncle}(x, \text{Paul})$, ...)

About the Peano Arithmetic domain

- Try to add the axioms defining other predicate or functions (e.g. " $n \leq m$ " or " $m * n$ ", " n^m ")
- Add some ground atom or its negation to the KB (ex: $1 = S(0)$, $2 = S(1)$, ...)
- Try to solve some query by entailment (e.g. $3 + 2 = 5$, $2 * 3 = 6$, ...)

Outline

- 1 Generalities
- 2 Syntax and Semantics of FOL
 - Syntax
 - Semantics
 - Satisfiability, Validity, Entailment
- 3 Using FOL
 - FOL Agents
 - Example: The Wumpus World

Example: The Wumpus World

The FOL KB

- **Perception**: binary predicate $\text{Percept}([s, b, g, b, sc], t)$
 - (recall: perception is [Stench, Breeze, Glitter, Bump, Scream])
 - **Stench, Breeze, Glitter, Bump, Scream** constant symbols
 - time step t represented as integer
- Percepts imply facts about the current state.
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Breeze}, g, m, c], t) \rightarrow \text{Breeze}(t))$
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Null}, g, m, c], t) \rightarrow \neg \text{Breeze}(t))$
 - ...
- **Environment**:
 - **Square**: term (pair of integers): $[1, 2]$
 - **Adjacency**: binary predicate Adjacent :
$$\forall x, y, a, b. (\text{Adjacent}([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$$
 - **Position**: predicate $\text{At}(\text{Agent}, s, t)$, ex: $\text{At}(\text{Agent}, [1, 1], 1)$
 - Unique position: $\forall x, s_1, s_2, t. ((\text{At}(x, s_1, t) \wedge \text{At}(x, s_2, t)) \rightarrow s_1 = s_2)$
 - **Wumpus**: predicate $\text{Wumpus}(s)$, ex: $\text{Wumpus}([3, 1])$
 - **Pits**: predicate $\text{Pit}(s)$, ex: $\text{Pit}([3, 1])$

Example: The Wumpus World

The FOL KB

- **Perception**: binary predicate $\text{Percept}([s, b, g, b, sc], t)$
 - (recall: perception is [Stench, Breeze, Glitter, Bump, Scream])
 - **Stench, Breeze, Glitter, Bump, Scream** constant symbols
 - time step t represented as integer
- Percepts imply facts about the current state.
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Breeze}, g, m, c], t) \rightarrow \text{Breeze}(t))$
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Null}, g, m, c], t) \rightarrow \neg \text{Breeze}(t))$
 - ...
- **Environment**:
 - **Square**: term (pair of integers): $[1, 2]$
 - **Adjacency**: binary predicate **Adjacent**:
$$\forall x, y, a, b. (\text{Adjacent}([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$$
 - **Position**: predicate $\text{At}(\text{Agent}, s, t)$, ex: $\text{At}(\text{Agent}, [1, 1], 1)$
 - Unique position: $\forall x, s_1, s_2, t. ((\text{At}(x, s_1, t) \wedge \text{At}(x, s_2, t)) \rightarrow s_1 = s_2)$
 - **Wumpus**: predicate $\text{Wumpus}(s)$, ex: $\text{Wumpus}([3, 1])$
 - **Pits**: predicate $\text{Pit}(s)$, ex: $\text{Pit}([3, 1])$

Example: The Wumpus World

The FOL KB

- **Perception**: binary predicate $\text{Percept}([s, b, g, b, sc], t)$
 - (recall: perception is [Stench, Breeze, Glitter, Bump, Scream])
 - **Stench, Breeze, Glitter, Bump, Scream** constant symbols
 - time step t represented as integer
- Percepts imply facts about the current state.
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Breeze}, g, m, c], t) \rightarrow \text{Breeze}(t))$
 - $\forall t, s, g, m, c. (\text{Percept}([s, \text{Null}, g, m, c], t) \rightarrow \neg \text{Breeze}(t))$
 - ...
- **Environment**:
 - **Square**: term (pair of integers): $[1, 2]$
 - **Adjacency**: binary predicate **Adjacent**:
$$\forall x, y, a, b. (\text{Adjacent}([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$$
 - **Position**: predicate $\text{At}(\text{Agent}, s, t)$, ex: $\text{At}(\text{Agent}, [1, 1], 1)$
 - Unique position: $\forall x, s_1, s_2, t. ((\text{At}(x, s_1, t) \wedge \text{At}(x, s_2, t)) \rightarrow s_1 = s_2)$
 - **Wumpus**: predicate $\text{Wumpus}(s)$, ex: $\text{Wumpus}([3, 1])$
 - **Pits**: predicate $\text{Pit}(s)$, ex: $\text{Pit}([3, 1])$

- For Wumpus, AIMA suggests;
 - **Wumpus**: constant, ex $\forall t. At(Wumpus, [2, 2], t)$
- Simplification: assume Wumpus status does not evolve with time
 - predicate $Wumpus(s)$, ex: $Wumpus([3, 1])$
 - ⇒ makes inference much easier
 - if we consider the case the Wumpus is killed by arrow, then we need reintroducing the “At” formalization

- For Wumpus, AIMA suggests;
 - **Wumpus**: constant, ex $\forall t. At(Wumpus, [2, 2], t)$
- Simplification: assume Wumpus status does not evolve with time
 - predicate $Wumpus(s)$, ex: $Wumpus([3, 1])$
 - ⇒ makes inference much easier
 - if we consider the case the Wumpus is killed by arrow, then we need reintroducing the “At” formalization

Example: The Wumpus World [cont.]

The FOL KB [cont.]

- Infer properties from percepts:
 - $\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$
 - $\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$
- Infer information about pits & Wumpus
 - $\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$
 - $\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$
- Evolution on time: successor states:
 - $\forall t. (HaveArrow(t+1) \leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t)))$
- Actions: terms Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb
 - simple reflex action: $\forall t. (Glitter(t) \rightarrow BestAction(Grab, t))$
 - Query: $AskVars[\exists a. BestAction(a, 5)] \implies \{a/Grab\}$

Personal remark

Simplified action axiomatization: "Move(...)" instead of "Turn(...), Forward"

Example: The Wumpus World [cont.]

The FOL KB [cont.]

- Infer properties from percepts:
 - $\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$
 - $\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$
- Infer information about pits & Wumpus
 - $\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$
 - $\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$
- Evolution on time: successor states:
 - $\forall t. (HaveArrow(t+1) \leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t)))$
- Actions: terms Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb
 - simple reflex action: $\forall t. (Glitter(t) \rightarrow BestAction(Grab, t))$
 - Query: $AskVars[\exists a. BestAction(a, 5)] \implies \{a/Grab\}$

Personal remark

Simplified action axiomatization: "Move(...)" instead of "Turn(...), Forward"

Example: The Wumpus World [cont.]

The FOL KB [cont.]

- Infer properties from percepts:
 - $\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$
 - $\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$
- Infer information about pits & Wumpus
 - $\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$
 - $\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$
- Evolution on time: successor states:
 - $\forall t. (HaveArrow(t+1) \leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t)))$
- Actions: terms Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb
 - simple reflex action: $\forall t. (Glitter(t) \rightarrow BestAction(Grab, t))$
 - Query: $AskVars[\exists a. BestAction(a, 5)] \implies \{a/Grab\}$

Personal remark

Simplified action axiomatization: "Move(...)" instead of "Turn(...), Forward"

Example: The Wumpus World [cont.]

The FOL KB [cont.]

- Infer properties from percepts:
 - $\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$
 - $\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$
- Infer information about pits & Wumpus
 - $\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$
 - $\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$
- Evolution on time: successor states:
 - $\forall t. (HaveArrow(t+1) \leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t)))$
- **Actions:** terms **Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb**
 - simple reflex action: $\forall t. (Glitter(t) \rightarrow BestAction(Grab, t))$
 - Query: $AskVars[\exists a. BestAction(a, 5)] \Rightarrow \{a/Grab\}$

Personal remark

Simplified action axiomatization: "Move(...)" instead of "*Turn(...), Forward*"

Example: The Wumpus World [cont.]

The FOL KB [cont.]

- Infer properties from percepts:
 - $\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$
 - $\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$
- Infer information about pits & Wumpus
 - $\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$
 - $\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$
- Evolution on time: successor states:
 - $\forall t. (HaveArrow(t+1) \leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t)))$
- **Actions:** terms **Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb**
 - simple reflex action: $\forall t. (Glitter(t) \rightarrow BestAction(Grab, t))$
 - Query: $AskVars[\exists a. BestAction(a, 5)] \Rightarrow \{a/Grab\}$

Personal remark

Simplified action axiomatization: “Move(...)” instead of “Turn(...), Forward”

Example: Exploring the Wumpus World

KB initially contains:

$$\forall x, y, a, b. (Adjacent([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$$
$$\forall t, s, g, m, c. (Percept([s, Null, g, m, c], t) \rightarrow \neg Breeze(t))$$
$$\forall t, b, g, m, c. (Percept([Null, b, g, m, c], t) \rightarrow \neg Stench(t))$$
$$\forall s, t. ((At(Agent, s, t) \wedge \neg Breeze(t)) \rightarrow \neg Breezy(s))$$
$$\forall s, t. ((At(Agent, s, t) \wedge \neg Stench(t)) \rightarrow \neg Stenchy(s))$$
$$\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$$
$$\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$$
$$\forall s. (Ok(s) \leftrightarrow (\neg Pit(s) \wedge \neg Wumpus(s)))$$

- A is initially in 1,1: $At(A, [1, 1], 0)$

- Perceives no stench, no breeze:

$$Tell[KB, Percept([Null, Null, Null, Null, Null], 0)]$$
$$\Rightarrow \neg Breeze(0), \neg Stench(0),$$
$$\Rightarrow \neg Breezy([1, 1]), \neg Stenchy([1, 1]),$$
$$\Rightarrow \neg Pit([1, 2]), \neg Pit([2, 1] \neg Wumpus([1, 2]), \neg Wumpus([2, 1]),$$
$$\Rightarrow Ok([1, 2]), Ok([2, 1])$$
$$AskVars[KB, \exists a. BestAction(a, 0)]$$
$$\Rightarrow \{a/Move([1, 2])\}, \{a/Move([2, 1])\}$$

OK			
OK A	OK		

Example: Exploring the Wumpus World

KB initially contains:

$\neg Pit([1, 1]), \neg Wumpus([1, 1]), \dots$

$\forall x, y, a, b. (Adjacent([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$

$\forall t, s, g, m, c. (Percept([s, Breeze, g, m, c], t) \rightarrow Breeze(t))$

$\forall t, b, g, m, c. (Percept([Null, b, g, m, c], t) \rightarrow \neg Stench(t))$

$\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$

$\forall s, t. ((At(Agent, s, t) \wedge \neg Stench(t)) \rightarrow \neg Stenchy(s))$

$\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$

$\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$

- Agent moves to [2,1]: $At(A, [2, 1], 1)$

- Perceives a breeze and no stench:

$Tell[KB, Percept([Null, Breeze, Null, Null, Null], 1)]$

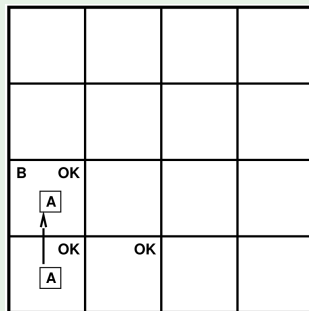
$\Rightarrow Breeze(1), \neg Stench(1),$

$\Rightarrow Breezy([2, 1]), \neg Stenchy([2, 1]),$

$\Rightarrow \exists r. (Adjacent(r, [2, 1]) \wedge Pit(r)),$
 $\neg Wumpus([3, 1]), \neg Wumpus([2, 2]),$

$\Rightarrow (Pit([3, 1]) \vee Pit([2, 2]))$

$AskVars[KB, \exists a. Action(a, 1)] \Rightarrow \{a/Move([1, 1])\}$



Example: Exploring the Wumpus World

KB initially contains:

$\neg Pit([1, 1]), \neg Wumpus([1, 1]), \dots$

$\forall x, y, a, b. (Adjacent([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$

$\forall t, s, g, m, c. (Percept([s, Breeze, g, m, c], t) \rightarrow Breeze(t))$

$\forall t, b, g, m, c. (Percept([Null, b, g, m, c], t) \rightarrow \neg Stench(t))$

$\forall s, t. ((At(Agent, s, t) \wedge Breeze(t)) \rightarrow Breezy(s))$

$\forall s, t. ((At(Agent, s, t) \wedge \neg Stench(t)) \rightarrow \neg Stenchy(s))$

$\forall s. (Breezy(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Pit(r)))$

$\forall s. (Stench(s) \leftrightarrow \exists r. (Adjacent(r, s) \wedge Wumpus(r)))$

- Agent moves to [2,1]: $At(A, [2, 1], 1)$

- Perceives a breeze and no stench:

$Tell[KB, Percept([Null, Breeze, Null, Null, Null], 1)]$

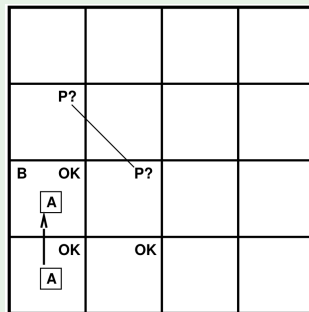
$\Rightarrow Breeze(1), \neg Stench(1),$

$\Rightarrow Breezy([2, 1]), \neg Stenchy([2, 1]),$

$\Rightarrow \exists r. (Adjacent(r, [2, 1]) \wedge Pit(r)),$
 $\neg Wumpus([3, 1]), \neg Wumpus([2, 2]),$

$\Rightarrow (Pit([3, 1]) \vee Pit([2, 2]))$

$AskVars[KB, \exists a. Action(a, 1)] \Rightarrow \{a / Move([1, 1])\}$



Exercise

Complete the example in the FOL case (see the PL case).