Fundamentals 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_2024/

Teaching assistants:

Mauro Dragoni, dragoni@fbk.eu, https://www.maurodragoni.com/teaching/fai/

Paolo Morettin, paolo.morettin@unitn.it, https://paolomorettin.github.io/

M.S. Course "Artificial Intelligence Systems", academic year 2024-2025

Last update: Thursday 5th September, 2024, 18:58

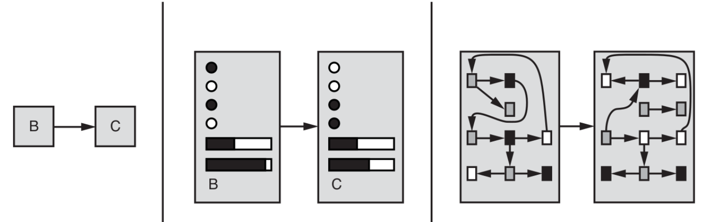
Outline

Outline

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



(a) Atomic (b) Factored (b) Structured

(© S. Russell & P. Norwig, AIMA)

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 $(A \land B) \rightarrow C$ 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

$\Longrightarrow$ 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, ...)
  - ...

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", ...

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”, ...

Outline

Outline

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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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 $\Longrightarrow$ 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: $\neg, \land, \lor, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- 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 $\Longrightarrow$ 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 $\Longrightarrow$ PL subcase of FOL
- Signature: the set of predicate, function & constant symbols

FOL: Syntax

- Terms:
  - constant or variable or $function(term_1, ..., term_n)$
  - ex: KingJohn, x, LeftLeg(Richard), (z*log(2))
  - denote objects in the real world (aka domain)
- Atomic sentences (aka atomic formulas):
  - $\top$, $\bot$
  - *proposition* or $predicate(term_1, ..., term_n)$ 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*$_1$, ..., *term*$_n$)
  - ex: KingJohn, x, LeftLeg(Richard), (z*log(2))
  - denote objects in the real world (aka domain)
- Atomic sentences (aka atomic formulas):
  - $\top$, $\bot$
  - *proposition* or *predicate*(*term*$_1$, ..., *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 $function(term_1, ..., term_n)$
  - ex: KingJohn, x, LeftLeg(Richard), (z*log(2))
  - denote objects in the real world (aka domain)
- Atomic sentences (aka atomic formulas):
  - $\top$, $\bot$
  - $proposition$ or $predicate(term_1, ..., term_n)$ 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 $function(term_1, ..., term_n)$
  - ex: KingJohn, x, LeftLeg(Richard), (z*log(2))
  - denote objects in the real world (aka domain)
- Atomic sentences (aka atomic formulas):
  - $\top$, $\bot$
  - $proposition$ or $predicate(term_1, ..., term_n)$ 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: $\forall x \exists y.(x > y)$)

$\Longrightarrow$ 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: $\forall x \exists y.(x > y)$)

$\Longrightarrow$ 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: $\forall x \exists y.(x > y)$)

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

FOL: Syntax (BNF)

$$
\begin{array}{lcl}
\langle\text{Sentence}\rangle & ::= & \langle\text{AtomicSentence}\rangle \mid \langle\text{ComplexSentence}\rangle \\
\langle\text{AtomicSentence}\rangle & ::= & \top \mid \bot \mid \\
 & & \langle\text{PredicateSymbol}\rangle(\langle\text{Term}\rangle, \ldots) \mid \\
 & & \langle\text{Term}\rangle = \langle\text{Term}\rangle \\
\langle\text{ComplexSentence}\rangle & ::= & \neg\langle\text{Sentence}\rangle \mid \\
 & & \langle\text{Sentence}\rangle\ \langle\text{Connective}\rangle\ \langle\text{Sentence}\rangle \mid \\
 & & \langle\text{Quantifier}\rangle\ \langle\text{Sentence}\rangle \\
\langle\text{Term}\rangle & ::= & \langle\text{ConstantSymbol}\rangle \mid \langle\text{Variable}\rangle \mid \\
 & & \langle\text{FunctionSymbol}\rangle(\langle\text{Term}\rangle, \ldots) \\
\langle\text{Connective}\rangle & ::= & \wedge \mid \vee \mid \rightarrow \mid \leftarrow \mid \leftrightarrow \mid \oplus \\
\langle\text{Quantifier}\rangle & ::= & \forall\ \langle\text{Variable}\rangle. \mid \exists\ \langle\text{Variable}\rangle. \\
\langle\text{Variable}\rangle & ::= & a \mid b \mid \cdots \mid x \mid y \mid \cdots \\
\langle\text{ConstantSymbol}\rangle & ::= & A \mid B \mid \cdots \mid John \mid 0 \mid 1 \mid \cdots \mid \pi \mid \ldots \\
\langle\text{FunctionSymbol}\rangle & ::= & F \mid G \mid \cdots \mid Cos \mid FatherOf \mid + \mid \ldots \\
\langle\text{PredicateSymbol}\rangle & ::= & P \mid Q \mid \cdots \mid Red \mid Brother \mid > \mid \cdots
\end{array}
$$

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

Outline

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 $P(t_1, ..., t_n)$ is true in an interpretation iff the objects referred to by $t_1, ..., t_n$ 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 $\geq$ 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 $P(t_1, ..., t_n)$ is true in an interpretation iff the objects referred to by $t_1, ..., t_n$ 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 $P(t_1, ..., t_n)$ is true in an interpretation iff the objects referred to by $t_1, ..., t_n$ 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 $P(t_1, ..., t_n)$ is true in an interpretation iff the objects referred to by $t_1, ..., t_n$ 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 $\longmapsto$ domain elements:
    a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
  - predicate symbols $\longmapsto$ domain relations:
    a k-ary predicate $P(...)$ 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 $\longmapsto$ domain functions:
    a k-ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \longmapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)

  (we denote by $[.]^{\mathcal{I}}$ 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 $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle$*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 $\longmapsto$ domain elements:
    a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
  - predicate symbols $\longmapsto$ domain relations:
    a k-ary predicate $P(...)$ 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 $\longmapsto$ domain functions:
    a k-ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \longmapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)

  (we denote by $[.]^{\mathcal{I}}$ 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 $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle$*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 $\longmapsto$ domain elements:
    a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
  - predicate symbols $\longmapsto$ domain relations:
    a k-ary predicate $P(...)$ 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 $\longmapsto$ domain functions:
    a k-ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \longmapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)

(we denote by $[.]^{\mathcal{I}}$ 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 $\langle \mathcal{D}, \mathcal{I} \rangle$ ($\langle$*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 $\longmapsto$ domain elements:
    a constant symbol C is mapped into a particular object $[C]^{\mathcal{I}}$ in $\mathcal{D}$
  - predicate symbols $\longmapsto$ domain relations:
    a k-ary predicate $P(...)$ 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 $\longmapsto$ domain functions:
    a k-ary function f is mapped into a domain function $[f]^{\mathcal{I}} : \mathcal{D}^k \longmapsto \mathcal{D}$ ($[f]^{\mathcal{I}}$ must be total)

(we denote by $[.]^{\mathcal{I}}$ 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 $\longmapsto$ domain elements:
    a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, ..., t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, ..., t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, ..., t_k$:
  - $[f(t_1, ..., t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}})$
  - Ex: if "Me, Mother, Father" are interpreted as usual, then "Mother(Father(Me))" is interpreted as my (paternal) grandmother
  - Ex: if "$+, -, \cdot, 0, 1, 2, 3, 4$" 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 $\longmapsto$ domain elements:
    a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, ..., t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, ..., t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, ..., t_k$:
  - $[f(t_1, ..., t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}})$
  - Ex: if "Me, Mother, Father" are interpreted as usual, then "Mother(Father(Me))" is interpreted as my (paternal) grandmother
  - Ex: if "$+, -, \cdot, 0, 1, 2, 3, 4$" 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 $\longmapsto$ domain elements:
    a variable x is mapped into a particular object $[x]^{\mathcal{I}}$ in $\mathcal{D}$
- A term $f(t_1, ..., t_k)$ is mapped by $\mathcal{I}$ into the value $[f(t_1, ..., t_k)]^{\mathcal{I}}$ returned by applying the domain function $[f]^{\mathcal{I}}$, into which f is mapped, to the values $[t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}}$ obtained by applying recursively $\mathcal{I}$ to the terms $t_1, ..., t_k$:
  - $[f(t_1, ..., t_k)]^{\mathcal{I}} = [f]^{\mathcal{I}}([t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}})$
  - Ex: if "Me, Mother, Father" are interpreted as usual, then "Mother(Father(Me))" is interpreted as my (paternal) grandmother
  - Ex: if "$+, -, \cdot, 0, 1, 2, 3, 4$" 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 $P(t_1, ..., t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms t_1,...t_k are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
  - $[P(t_1, ..., t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
  - Ex: if "Me, Mother, Father, Married" are interpreted as traditon, 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 $P(t_1, ..., t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms t_1,...t_k are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
  - $[P(t_1, ..., t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
  - Ex: if "Me, Mother, Father, Married" are interpreted as traditon, 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 $P(t_1, ..., t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms t_1,...t_k are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
  - $[P(t_1, ..., t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
  - Ex: if "Me, Mother, Father, Married" are interpreted as traditon, 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

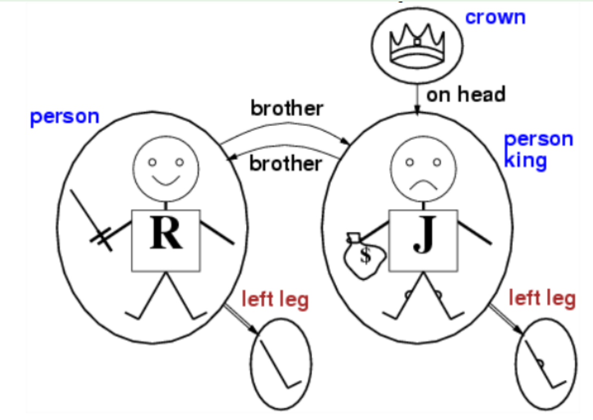
$\mathcal{I}$ maps formulas into truth values

- An atomic formula $P(t_1, ..., t_k)$ is true in $\mathcal{I}$ iff the objects into which the terms t_1,...t_k are mapped by $\mathcal{I}$ comply to the relation into which P is mapped
  - $[P(t_1, ..., t_k)]^{\mathcal{I}}$ is true iff $\langle [t_1]^{\mathcal{I}}, ..., [t_k]^{\mathcal{I}} \rangle \in [P]^{\mathcal{I}}$
  - Ex: if "Me, Mother, Father, Married" are interpreted as traditon, 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 Lionhearth and John Lackland

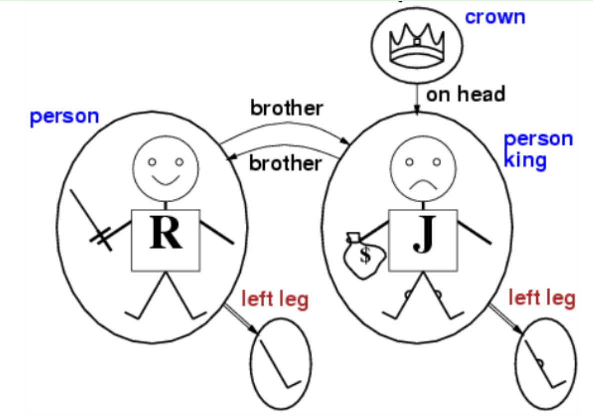
- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
  - $[Richard]^{\mathcal{I}}$: Richard the Lionhearth
  - $[John]^{\mathcal{I}}$: evil King John
  - $[Brother]^{\mathcal{I}}$: brotherhood
- $[Brother(Richard, John)]^{\mathcal{I}}$ is true
- $[LeftLeg]^{\mathcal{I}}$ maps any individual to his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



Models for FOL: Example

Richard Lionhearth and John Lackland

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

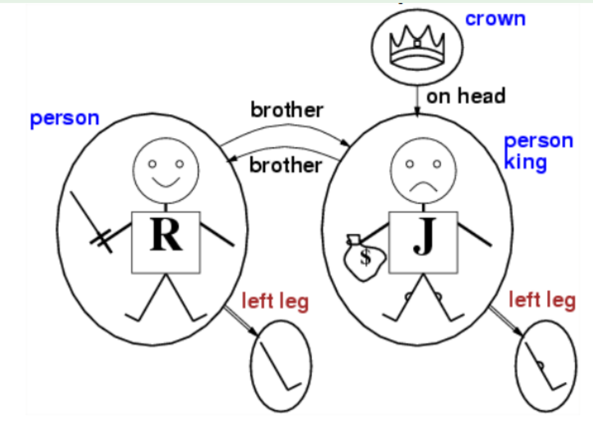


(© S. Russell & P. Norwig, AIMA)

Models for FOL: Example

Richard Lionhearth and John Lackland

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

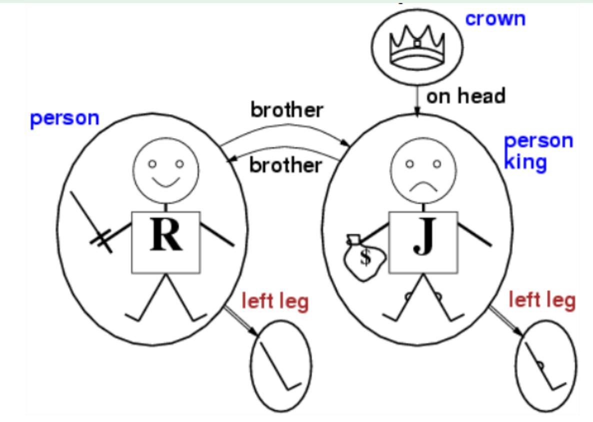


(© S. Russell & P. Norwig, AIMA)

Models for FOL: Example

Richard Lionhearth and John Lackland

- $\mathcal{D}$: domain at right
- $\mathcal{I}$: s.t.
  - $[Richard]^{\mathcal{I}}$: Richard the Lionhearth
  - $[John]^{\mathcal{I}}$: 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

- $\forall x.\alpha(x, ...)$ (x variable, typically occurs in x)
  - ex: $\forall x.(King(x) \rightarrow Person(x))$ ("all kings are persons")
- $\forall x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(John) & \rightarrow Person(John) &)\wedge \\
(King(Richard) & \rightarrow Person(Richard) &)\wedge \\
(King(crown) & \rightarrow Person(crown) &)\wedge \\
(King(LeftLeg(John)) & \rightarrow Person(LeftLeg(John)) &)\wedge \\
(King(LeftLeg(LeftLeg(John))) & \rightarrow Person(LeftLeg(LeftLeg(John))) &)\wedge \\
\ldots & \ldots &
\end{array}
$$

Universal Quantification

- $\forall x.\alpha(x, ...)$ (x variable, typically occurs in x)
  - ex: $\forall x.(King(x) \rightarrow Person(x))$ ("all kings are persons")
- $\forall x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(John) & \rightarrow Person(John) &)\wedge \\
(King(Richard) & \rightarrow Person(Richard) &)\wedge \\
(King(crown) & \rightarrow Person(crown) &)\wedge \\
(King(LeftLeg(John)) & \rightarrow Person(LeftLeg(John)) &)\wedge \\
(King(LeftLeg(LeftLeg(John))) & \rightarrow Person(LeftLeg(LeftLeg(John))) &)\wedge \\
\ldots & \ldots &
\end{array}
$$

Universal Quantification

- $\forall x.\alpha(x, ...)$ (x variable, typically occurs in x)
  - ex: $\forall x.(King(x) \rightarrow Person(x))$ ("all kings are persons")
- $\forall x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(John) & \rightarrow Person(John) &)\wedge \\
(King(Richard) & \rightarrow Person(Richard) &)\wedge \\
(King(crown) & \rightarrow Person(crown) &)\wedge \\
(King(LeftLeg(John)) & \rightarrow Person(LeftLeg(John)) &)\wedge \\
(King(LeftLeg(LeftLeg(John))) & \rightarrow Person(LeftLeg(LeftLeg(John))) &)\wedge \\
\ldots & \ldots &
\end{array}
$$

Universal Quantification

- $\forall x.\alpha(x, ...)$ (x variable, typically occurs in x)
  - ex: $\forall x.(King(x) \rightarrow Person(x))$ ("all kings are persons")
- $\forall x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(John) & \rightarrow Person(John) &)\wedge \\
(King(Richard) & \rightarrow Person(Richard) &)\wedge \\
(King(crown) & \rightarrow Person(crown) &)\wedge \\
(King(LeftLeg(John)) & \rightarrow Person(LeftLeg(John)) &)\wedge \\
(King(LeftLeg(LeftLeg(John))) & \rightarrow Person(LeftLeg(LeftLeg(John))) &)\wedge \\
... & ... &
\end{array}
$$

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: $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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: $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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:
  $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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: $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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: $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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: $\forall x.(P(x) \rightarrow \alpha(x, ...))$
  - ex "$\forall x.(King(x) \rightarrow ...)$", "$\forall x.(Integer(x) \rightarrow ...)$",
- 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

- $\exists x.\alpha(x, ...)$ (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 ..."
- $\exists x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(Richard) & \wedge Evil(Richard) &)\vee \\
(King(John) & \wedge Evil(John) &)\vee \\
(King(crown) & \wedge Evil(crown) &)\vee \\
(King(LeftLeg(John)) & \wedge Evil(LeftLeg(John)) &)\vee \\
(King(LeftLeg(LeftLeg(John))) & \wedge Evil(LeftLeg(LeftLeg(John))) &)\vee \\
... & ... &
\end{array}
$$

Existential Quantification

- $\exists x.\alpha(x, ...)$ (x variable, typically occurs in x)
  - ex: $\exists x.(King(x) \land Evil(x))$ ("there is an evil king")
  - pronounced "exists x s.t. ..." or "for some x ..."
- $\exists x.\alpha(x, ...)$ 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

$$
\begin{array}{lll}
(King(Richard) & \land Evil(Richard) &)\lor \\
(King(John) & \land Evil(John) &)\lor \\
(King(crown) & \land Evil(crown) &)\lor \\
(King(LeftLeg(John)) & \land Evil(LeftLeg(John)) &)\lor \\
(King(LeftLeg(LeftLeg(John))) & \land Evil(LeftLeg(LeftLeg(John))) &)\lor \\
\ldots & \ldots &
\end{array}
$$

Existential Quantification

- $\exists x.\alpha(x,...)$ (x variable, typically occurs in x)
  - ex: $\exists x.(King(x) \land Evil(x))$ ("there is an evil king")
  - pronounced "exists x s.t. ..." or "for some x ..."
- $\exists x.\alpha(x,...)$ 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

$$
\begin{array}{lll}
(King(Richard) & \land Evil(Richard) &)\lor \\
(King(John) & \land Evil(John) &)\lor \\
(King(crown) & \land Evil(crown) &)\lor \\
(King(LeftLeg(John)) & \land Evil(LeftLeg(John)) &)\lor \\
(King(LeftLeg(LeftLeg(John))) & \land Evil(LeftLeg(LeftLeg(John))) &)\lor \\
... & ... &
\end{array}
$$

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: $\exists x.(P(x) \wedge \alpha(x, ...))$
  - ex "$\exists x.(King(x) \wedge ...)$", "$\exists x.(Integer(x) \wedge ...)$",
- 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: $\exists x.(P(x) \wedge \alpha(x, ...))$
  - ex "$\exists x.(King(x) \wedge ...)$", "$\exists x.(Integer(x) \wedge ...)$",
- 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:
  $\exists x.(P(x) \wedge \alpha(x, ...))$
  - ex "$\exists x.(King(x) \wedge ...)$", "$\exists x.(Integer(x) \wedge ...)$",
- 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: $\exists x.(P(x) \land \alpha(x, ...))$
  - ex "$\exists x.(King(x) \land ...)$", "$\exists x.(Integer(x) \land ...)$",
- Beware of typical mistake: do not use "$\rightarrow$" instead of "$\land$"
  - ex: "$\exists x.(King(x) \rightarrow Evil(x))$" means "Someone is not a king or is evil"
  - ex: "$\exists x.(King(x) \land Evil(x))$" means "Someone is king and is evil" (i.e., "Some king is evil")
- "$\exists$" distributes with "$\lor$", but not with "$\land$"
  - $\exists x.(P(x) \lor Q(x))$ equivalent to $(\exists x.P(x)) \lor (\exists x.Q(x))$
  - "Somebody is a king or is a knight" same as "Somebody is a king or somebody is a knight"
  - $\exists x.(P(x) \land Q(x))$ not equivalent to $(\exists x.P(x)) \land (\exists x.Q(x))$
  - "Somebody is a king and is evil" much stronger than "Somebody is a king and somebody is evil"
    $\exists x.(P(x) \land Q(x)) \models (\exists x.P(x)) \land (\exists x.Q(x))$
    $(\exists x.P(x)) \land (\exists x.Q(x)) \not\models \exists x.(P(x) \land Q(x))$

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: $\exists x.(P(x) \wedge \alpha(x, ...))$
  - ex "$\exists x.(King(x) \wedge ...)$", "$\exists x.(Integer(x) \wedge ...)$",
- 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: $\exists x.(P(x) \wedge \alpha(x, ...))$
  - ex "$\exists x.(King(x) \wedge ...)$", "$\exists x.(Integer(x) \wedge ...)$",
- 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
  - $\forall x, y.\ (Mother(x, y) \leftrightarrow (Female(x) \land Parent(x, y)))$
- A first cousin is a child of a parent's sibling
  - $\forall x_1, x_2.\ (FirstCousin(x_1, x_2) \leftrightarrow$
    $\exists p_1, p_2.\ (Siblings(p_1, p_2) \land Parent(p_1, x_1) \land Parent(p_2, x_2)))$
- 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: $1 = 2$ and $x * x = x$ are satisfiable (!)
  - Ex: $2 = 2$ is valid
- Ex: definition of *Sibling* in terms of *Parent*
  $\forall x, y.\ (Siblings(x, y) \leftrightarrow [\neg(x = y) \land \exists p_1, p_2.\ (\neg(p_1 = p_2) \land Parent(p_1, x) \land Parent(p_2, x) \land Parent(p_1, y) \land Parent(p_2, y)]))$

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: $1 = 2$ and $x * x = x$ are satisfiable (!)
  - Ex: $2 = 2$ is valid
- Ex: definition of *Sibling* in terms of *Parent*

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

Example

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

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

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., $\forall x.\exists yP(x,y)$ 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., $\forall x.\exists y P(x, y)$ 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 $\forall y.\exists yP(y, y)$!

Properties of Quantifiers

Notation variants: $\forall x(\forall y.\alpha) \Longleftrightarrow \forall x \forall y.\alpha \Longleftrightarrow \forall x, y.\alpha \Longleftrightarrow \forall xy.\alpha$
(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 $\forall y.\exists yP(y, y)$!

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 "$\land$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \land \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \land \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \land \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\land, \lor\rangle$ 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
  - $\forall x.\mathit{Likes}(x, \mathit{Icecream})$ equivalent to $\neg\exists x.\neg\mathit{Likes}(x, \mathit{Icecream})$
  - $\exists x.\mathit{Likes}(x, \mathit{Broccoli})$ equivalent to $\neg\forall x.\neg\mathit{Likes}(x, \mathit{Broccoli})$
- 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"
  - $\neg\forall x.(\mathit{King}(x) \rightarrow \mathit{Evil}(x)) \iff \exists x.(\mathit{King}(x) \wedge \neg\mathit{Evil}(x))$
- 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

Outline

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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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 $\mathcal{M}$, $\mathcal{M} \models \alpha \Longrightarrow \mathcal{M} \models \beta$ (i.e., $M(\alpha) \subseteq M(\beta)$)
- φ is valid ($\models \varphi$) iff $\mathcal{M} \models \varphi$ forall $\mathcal{M}$s (i.e., $\mathcal{M} \in M(\varphi)$ forall $\mathcal{M}$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 $\Gamma \stackrel{\text{def}}{=} \{\varphi_1, ..., \varphi_n\}$. 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
- $P(x) \land \neg P(x)$, $\neg(x = x)$, $(\forall x, y.Q(x, y)) \rightarrow \neg Q(a, b)$ 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
- $\forall x.P(x) \land \neg\exists x.P(x))$ 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
- $P(x) \land \neg P(x)$, $\neg(x = x)$, $(\forall x, y.Q(x, y)) \rightarrow \neg Q(a, b)$ 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
- $\forall x.P(x) \land \neg\exists x.P(x))$ 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
                  ...
- $\Longrightarrow$ 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
                  ...
- $\Longrightarrow$ 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

©Munch Museum, Oslo

Outline

Outline

[Recall:] Knowledge-Based Agent: General Schema

- Given a percept, the agent
  - Tells the KB of the percept at time step t
  - ASKs the KB for the best action to do at time step t
  - Tells the KB that it has in fact taken that action
- Details hidden in three functions: MAKE-PERCEPT-SENTENCE, MAKE-ACTION-QUERY, MAKE-ACTION-SENTENCE
  - construct logic sentences
  - implement the interface between sensors/actuators and KRR core
- Tell and Ask may require complex logical inference

```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

FOL Knowledge-Based Agent

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

$\Longrightarrow$ Ask(KB,α) returns true only if $KB \models \alpha$

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

FOL Knowledge-Based Agent

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

$\Longrightarrow$ Ask(KB,α) returns true only if $KB \models \alpha$

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

FOL Knowledge-Based Agent

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

$\Longrightarrow$ Ask(KB,α) returns true only if $KB \models \alpha$

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

FOL Knowledge-Based Agent

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

$\Longrightarrow$ Ask(KB,α) returns true only if $KB \models \alpha$

- Other queries: AskVars, asking for variable values
  $\Longrightarrow$ returns one (or more) binding lists (aka substitutions) $\{var/term; var/term, ...\}$
  - ex: $\text{AskVars}(KB, \exists x.Person(x)) \Longrightarrow \{x/John\}; \{x/Richard\}$
  - typical for Horn clauses
    (e.g. with $King(John) \vee King(Richard)$,
    the query $\text{AskVars}(KB, \exists x.King(x))$ 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. $\forall x, y.(x = Mother(y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
  2. $\forall x, y.(Brother(x, y) \leftrightarrow (Male(x) \wedge Sibling(x, y)))$
  3. $\forall x, y.(Grandparent(x, y) \leftrightarrow \exists z.(Parent(x, z) \wedge Parent(z, y)))$
  4. $\forall x, y.(Sibling(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2.((p_1 \neq p_2)\wedge$ $Parent(p_1, x) \wedge Parent(p_1, y) \wedge (Parent(p_2, x) \wedge Parent(p_2, y))))$
  5. ...
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.(Sibling(x, y) \leftrightarrow Sibling(y, x))$

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. $\forall x, y.(x = Mother(y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
  2. $\forall x, y.(Brother(x, y) \leftrightarrow (Male(x) \wedge Sibling(x, y)))$
  3. $\forall x, y.(Grandparent(x, y) \leftrightarrow \exists z.(Parent(x, z) \wedge Parent(z, y)))$
  4. $\forall x, y.(Sibling(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2.((p_1 \neq p_2) \wedge$ $Parent(p_1, x) \wedge Parent(p_1, y) \wedge (Parent(p_2, x) \wedge Parent(p_2, y))))$
  5. ...
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.(Sibling(x, y) \leftrightarrow Sibling(y, x))$

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. $\forall x, y.(x = Mother(y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
  2. $\forall x, y.(Brother(x, y) \leftrightarrow (Male(x) \wedge Sibling(x, y)))$
  3. $\forall x, y.(Grandparent(x, y) \leftrightarrow \exists z.(Parent(x, z) \wedge Parent(z, y)))$
  4. $\forall x, y.(Sibling(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2.((p_1 \neq p_2) \wedge Parent(p_1, x) \wedge Parent(p_1, y) \wedge (Parent(p_2, x) \wedge Parent(p_2, y))))$
  5. ...
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.(Sibling(x, y) \leftrightarrow Sibling(y, x))$

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. $\forall x, y.(x = Mother(y) \leftrightarrow (Female(x) \wedge Parent(x, y)))$
  2. $\forall x, y.(Brother(x, y) \leftrightarrow (Male(x) \wedge Sibling(x, y)))$
  3. $\forall x, y.(Grandparent(x, y) \leftrightarrow \exists z.(Parent(x, z) \wedge Parent(z, y)))$
  4. $\forall x, y.(Sibling(x, y) \leftrightarrow ((x \neq y) \wedge \exists p_1, p_2.((p_1 \neq p_2) \wedge Parent(p_1, x) \wedge Parent(p_1, y) \wedge (Parent(p_2, x) \wedge Parent(p_2, y))))$
  5. ...
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.(Sibling(x, y) \leftrightarrow Sibling(y, x))$

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,...
- Knowledge base KB:
  1. $NatNum(0)$
  2. $\forall x.(NatNum(x) \rightarrow NatNum(S(x)))$
  3. $\forall x.(NatNum(x) \rightarrow (0 \neq S(x)))$
  4. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
  5. $\forall x.(NatNum(x) \rightarrow (x = (0 + x)))$
  6. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow (S(x) + y) = S(x + y))$
  7. $1 = S(0), 2 = S(1), 3 = S(2), \ldots$
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.((NatNum(x) \wedge (NatNum(y))) \rightarrow ((x + y) = (y + x)))$

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,...
- Knowledge base KB:
  1. $NatNum(0)$
  2. $\forall x.(NatNum(x) \rightarrow NatNum(S(x)))$
  3. $\forall x.(NatNum(x) \rightarrow (0 \neq S(x)))$
  4. $\forall x, y.((NatNum(x) \land NatNum(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
  5. $\forall x.(NatNum(x) \rightarrow (x = (0 + x)))$
  6. $\forall x, y.((NatNum(x) \land NatNum(y)) \rightarrow (S(x) + y) = S(x + y))$
  7. $1 = S(0), 2 = S(1), 3 = S(2), \ldots$
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.((NatNum(x) \land (NatNum(y))) \rightarrow ((x + y) = (y + x)))$

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,...
- Knowledge base KB:
  1. $NatNum(0)$
  2. $\forall x.(NatNum(x) \rightarrow NatNum(S(x)))$
  3. $\forall x.(NatNum(x) \rightarrow (0 \neq S(x)))$
  4. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
  5. $\forall x.(NatNum(x) \rightarrow (x = (0 + x)))$
  6. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow (S(x) + y) = S(x + y))$
  7. $1 = S(0)$, $2 = S(1)$, $3 = S(2)$, ...
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.((NatNum(x) \wedge (NatNum(y))) \rightarrow ((x + y) = (y + x)))$

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,...
- Knowledge base KB:
  1. $NatNum(0)$
  2. $\forall x.(NatNum(x) \rightarrow NatNum(S(x)))$
  3. $\forall x.(NatNum(x) \rightarrow (0 \neq S(x)))$
  4. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
  5. $\forall x.(NatNum(x) \rightarrow (x = (0 + x)))$
  6. $\forall x, y.((NatNum(x) \wedge NatNum(y)) \rightarrow (S(x) + y) = S(x + y))$
  7. $1 = S(0), 2 = S(1), 3 = S(2), \ldots$
- Queries inferred from KB
  - ex: (4) $\models \forall x, y.((NatNum(x) \wedge (NatNum(y))) \rightarrow ((x + y) = (y + x)))$

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), $\exists x.Uncle(x, Paul)$, ...)

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), $\exists x.Uncle(x, Paul)$, ...)

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

Example: The Wumpus World

The FOL KB

- Perception: binary predicate 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.
  - $\forall t, s, g, m, c.(Percept([s, Breeze, g, m, c], t) \rightarrow Breeze(t))$
  - $\forall t, s, g, m, c.(Percept([s, Null, g, m, c], t) \rightarrow \neg Breeze(t))$
  - ...
- Environment:
  - Square: term (pair of integers): $[1, 2]$
  - Adjacency: binary predicate Adjacent:
    $$\forall x, y, a, b.(Adjacent([x, y], [a, b]) \leftrightarrow (x = a \land (y = b - 1 \lor y = b + 1)) \lor (y = b \land (x = a - 1 \lor x = a + 1)))$$
  - Position: predicate $At(Agent, s, t)$, ex: $At(Agent, [1, 1], 1)$
  - Unique position: $\forall x, s_1, s_2, t.((At(x, s_1, t) \land At(x, s_2, t)) \rightarrow s_1 = s_2)$
  - Wumpus: predicate $Wumpus(s)$, ex: $Wumpus([3, 1])$
  - Pits: predicate $Pit(s)$, ex: $Pit([3, 1])$

Example: The Wumpus World

The FOL KB

- Perception: binary predicate 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.
  - $\forall t, s, g, m, c.(Percept([s, Breeze, g, m, c], t) \rightarrow Breeze(t))$
  - $\forall t, s, g, m, c.(Percept([s, Null, g, m, c], t) \rightarrow \neg Breeze(t))$
  - ...
- Environment:
  - Square: term (pair of integers): $[1, 2]$
  - Adjacency: binary predicate Adjacent:
    $\forall x, y, a, b.(Adjacent([x, y], [a, b]) \leftrightarrow$
    $(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$
  - Position: predicate $At(Agent, s, t)$, ex: $At(Agent, [1, 1], 1)$
  - Unique position: $\forall x, s_1, s_2, t.((At(x, s_1, t) \wedge At(x, s_2, t)) \rightarrow s_1 = s_2)$
  - Wumpus: predicate $Wumpus(s)$, ex: $Wumpus([3, 1])$
  - Pits: predicate $Pit(s)$, ex: $Pit([3, 1])$

Example: The Wumpus World

The FOL KB

- Perception: binary predicate 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.
  - $\forall t, s, g, m, c.(Percept([s, Breeze, g, m, c], t) \rightarrow Breeze(t))$
  - $\forall t, s, g, m, c.(Percept([s, Null, g, m, c], t) \rightarrow \neg Breeze(t))$
  - ...
- Environment:
  - Square: term (pair of integers): $[1, 2]$
  - Adjacency: binary predicate Adjacent:
    $$\forall x, y, a, b.(Adjacent([x, y], [a, b]) \leftrightarrow (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)))$$
  - Position: predicate $At(Agent, s, t)$, ex: $At(Agent, [1, 1], 1)$
  - Unique position: $\forall x, s_1, s_2, t.((At(x, s_1, t) \wedge At(x, s_2, t)) \rightarrow s_1 = s_2)$
  - Wumpus: predicate $Wumpus(s)$, ex: $Wumpus([3, 1])$
  - Pits: predicate $Pit(s)$, ex: $Pit([3, 1])$

Personal Remark

- 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: $Wumpus([3, 1])$

  $\Longrightarrow$ makes inference much easier
  - if we consider the case the Wumpus is killed by arrow, then we need reintroducing the “At” formalization

Personal Remark

- 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: $Wumpus([3,1])$

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

Personal remark

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

Example: The Wumpus World [cont.]

The FOL KB [cont.]

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

Personal remark

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

Example: The Wumpus World [cont.]

The FOL KB [cont.]

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

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: $AskVars(\exists a.BestAction(a, 5)) \Longrightarrow \{a/Grab\}$

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: $AskVars(\exists a.BestAction(a, 5)) \Longrightarrow \{a/Grab\}$

Personal remark

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

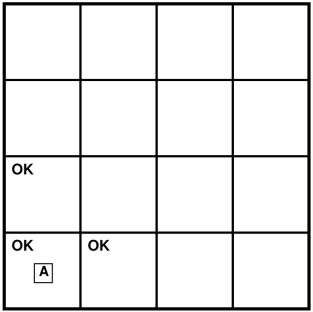
Example: Exploring the Wumpus World

KB initially contains:

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

- A is initially in 1,1: $At(A, [1, 1], 0)$
- Perceives no stench, no breeze:

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



Example: Exploring the Wumpus World

KB initially contains:

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

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

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

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

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

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

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

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

- Agent moves to [2,1]: $At(A, [2,1], 1)$
- Perceives a breeze and no stench:

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

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

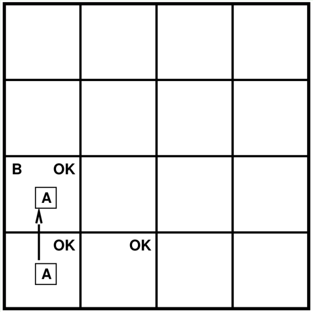
  $\Longrightarrow Breezy([2,1]), \neg Stenchy([2,1]),$

  $\Longrightarrow \exists r.(Adjacent(r, [2,1]) \wedge Pit(r)),$

  $\neg Wumpus([3,1]), \neg Wumpus([2,2]),$

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

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



Example: Exploring the Wumpus World

KB initially contains:

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

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

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

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

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

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

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

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

- Agent moves to [2,1]: $At(A, [2,1], 1)$
- Perceives a breeze and no stench:

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

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

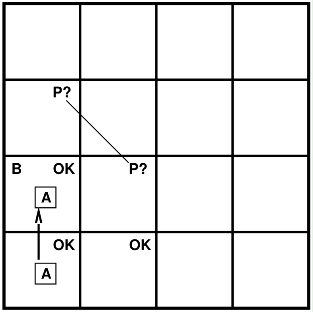
  $\Longrightarrow Breezy([2,1]), \neg Stenchy([2,1]),$

  $\Longrightarrow \exists r.(Adjacent(r, [2,1]) \wedge Pit(r)),$

  $\neg Wumpus([3,1]), \neg Wumpus([2,2]),$

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

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



Exercise

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