Formal Methods: Module I: Automated Reasoning Ch. 02: **Reasoning in First-Order Logic**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: http://disi.unitn.it/rseba/DIDATTICA/fm2021/
Teaching assistant: Giuseppe Spallitta – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2020-2021

last update: Tuesday 13th April, 2021, 13:55

Outline

Outline

1. First-Order Logic
   - Generalities
   - Syntax
   - Semantics
   - Satisfiability, Validity, Entailment
2. Basic First-Order Reasoning
   - Substitutions & Instantiations
   - From Propositional to First-Order Reasoning
   - Unification and Lifting
3. Resolution-based First-Order Reasoning
   - CNF-Ization
   - Resolution
   - Dealing with Equalities
   - A Complete Example

Outline

A Brief History of Logical Reasoning

When	Who	What
322 B.C.	Aristotle	"Syllogisms" (inference rules), quantifiers
1867	Boole	Propositional Logic
1879	Frege	First-Order Logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for PL (DP/DPLL)
1965	Robinson	"practical" algorithm for FOL (resolution)

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)
  - temporal logics (TLs)
  - ...

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)
  - temporal logics (TLs)
  - ...

Limits of Propositional Logic

Limits of Propositional Logic

- Is "Atomic": based on atomic events which cannot be decomposed
- Has very limited expressive power
  - 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

$\Longrightarrow$ cannot concisely describe an environment with many objects

Limits of Propositional Logic

Limits of Propositional Logic

- Is "Atomic": based on atomic events which cannot be decomposed
- Has very limited expressive power
  - 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

$\Longrightarrow$ cannot concisely describe an environment with many objects

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

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: $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$
- 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

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:
  - $\neg\alpha, \alpha \land \beta, \alpha \lor \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta, \alpha \oplus \beta$,
    $\forall x.\alpha, \exists x.\alpha$ s.t. x (typically) occurs in α
  - Ex: $\forall y.(Italian(y) \rightarrow President(Mattarella, y))$
    $\exists x \forall y.President(x, y) \rightarrow \forall y \exists x.President(x, y)$
    $\forall x.(P(x) \land Q(x)) \leftrightarrow ((\forall x.P(x)) \land (\forall x.Q(x)))$
    $\forall x.(((x \geq 0) \land (x \leq \pi)) \rightarrow (sin(x) \geq 0))$
  - 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 are quantified (ex: $\forall x \exists y.(x > y)$)

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 are quantified (ex: $\forall x \exists y.(x > y)$)

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 ≥ 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 $P(t_1, \ldots, t_n)$ is true in an interpretation iff the objects referred to by $t_1, \ldots, t_n$ 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

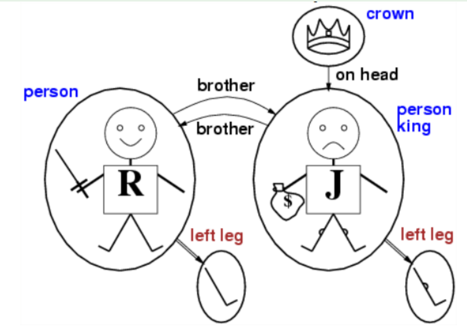
$\mathcal{I}$ maps formulas into truth values

- 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
- ...

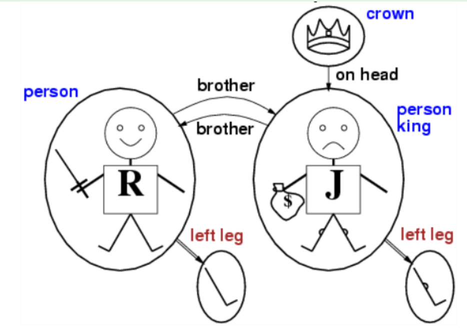


(© S. Russell & P. Norwig, AIMA)

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
- ...

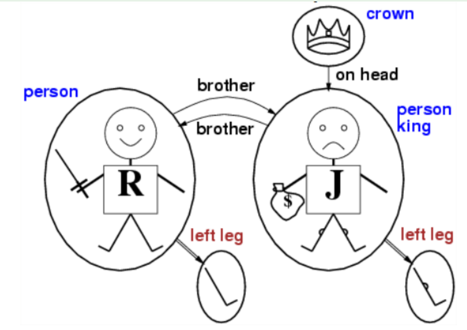


(© S. Russell & P. Norwig, AIMA)

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
- ...

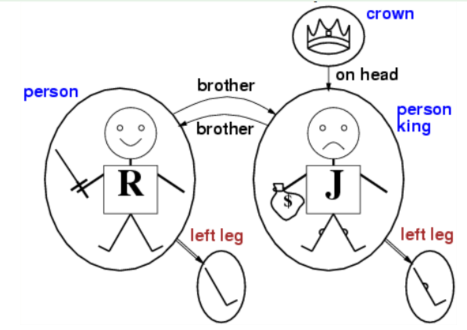


(© S. Russell & P. Norwig, AIMA)

Models for FOL: 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 ($[\![LeftLeg(crown) = null]\!]^{\mathcal{I}}$

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

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

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

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

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

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

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"
- "$\forall$" 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"

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"
- "$\forall$" 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"

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: $\forall x.(P(x) \to \alpha(x, ...))$
  - ex "$\forall x.(King(x) \to ...)$", "$\forall x.(Integer(x) \to ...)$",
- Beware of typical mistake: do not use "$\wedge$" instead of "$\to$"
  - ex: "$\forall x.(King(x) \wedge Person(x))$" means "everything/one is a King and 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"

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"
- "$\forall$" 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"

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 "$\land$" instead of "$\rightarrow$"
  - ex: "$\forall x.(King(x) \land Person(x))$" means "everything/one is a King and is a Person"
- "$\forall$" distributes with "$\land$", but not with "$\lor$"
  - $\forall x.(P(x) \land Q(x))$ equivalent to $(\forall x.P(x)) \land (\forall x.Q(x))$
  - "Everybody is a king and is a person" same as "Everybody is a king and everybody is a person"
  - $\forall x.(P(x) \lor Q(x))$ not equivalent to $(\forall x.P(x)) \lor (\forall x.Q(x))$
  - "Everybody is a king or is a peasant" much weaker than "Everybody is a king or everybody is a peasant"

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"
- "$\forall$" 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"

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

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

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

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

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 [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"
- "$\exists$" 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"

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"
- "$\exists$" 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"

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"
- "$\exists$" 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"

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"
- "$\exists$" 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"

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"
- "$\exists$" 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"

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"
- "$\exists$" 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"
  - $\exists x.(P(x) \land Q(x))$ <u>not</u> 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"

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
  - $\forall x, y.\ (Mother(x, y) \leftrightarrow (Female(x) \land Parent(x, y)))$
- 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
  - $\forall x, y.\ (Mother(x, y) \leftrightarrow (Female(x) \land Parent(x, y)))$
- 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
  - $\forall x.\ (Dog(x) \rightarrow Mammal(x))$

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*
  $\forall x, y.\ (Siblings(x, y) \leftrightarrow [\neg(x = y) \wedge \exists m, f.\ (\neg(m = f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]))$

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*
  $\forall x, y.\ (Siblings(x, y) \leftrightarrow [\neg(x = y) \land \exists m, f.\ (\neg(m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]))$

Example

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

Example

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

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

- 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
  - $\forall x, y.\ ((Sisters(x, y) \to (Female(x) \land Female(y))) \land (Brothers(x, y) \to (Male(x) \land Male(y))))$
- Every married person has a spouse
  - $\forall x.\ ((Person(x) \land Married(x)) \to \exists y.\ Spouse(x, y))$
- Married people have spouses
  - $\forall x.\ ((Person(x) \land Married(x)) \to \exists y.\ Spouse(x, y))$
- Only married people have spouses
  - $\forall x, y.\ ((Person(x) \land Person(y) \land Spouse(x, y)) \to (Married(x) \land Married(y)))$
- People cannot be married to their siblings
  - $\forall x, y.\ (Spouse(x, y) \to \neg Siblings(x, y))$

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
  - $\exists x.\ (Person(x) \land \neg\exists y.\ Spouse(x,y))$
- Everybody has a mother
  - $\forall x.\ (Person(x) \rightarrow \exists y.\ Mother(y,x))$
- Everybody has a mother and only one
  - $\forall x.\ Person(x) \rightarrow (\exists y.\ Mother(y,x) \land \neg\exists z.\ (\neg(y = z) \land Mother(z,x)))$

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

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"

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"

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"

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"

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 $\exists yx.Twins(x, y)$
- $\exists x \forall y.P(x, y)$ <u>not</u> 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"

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual
  - $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
  - $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
  - $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
  - $\neg\exists x.\alpha \iff \forall x.\neg\alpha$
- Examples
  - $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
  - $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$
- Negated restricted quantifiers switch "$\rightarrow$" with "$\wedge$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual
  - $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
  - $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
  - $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
  - $\neg\exists x.\alpha \iff \forall x.\neg\alpha$
- Examples
  - $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
  - $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$
- Negated restricted quantifiers switch "$\rightarrow$" with "$\wedge$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual
  - $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
  - $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
  - $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
  - $\neg\exists x.\alpha \iff \forall x.\neg\alpha$
- Examples
  - $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
  - $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$
- Negated restricted quantifiers switch "$\rightarrow$" with "$\wedge$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual
  - $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
  - $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
  - $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
  - $\neg\exists x.\alpha \iff \forall x.\neg\alpha$
- Examples
  - $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
  - $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$
- Negated restricted quantifiers switch "$\rightarrow$" with "$\wedge$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

Duality of Universal and Existential Quantification

- $\forall$ and $\exists$ are dual
  - $\forall x.\alpha \iff \neg\exists x.\neg\alpha$
  - $\neg\forall x.\alpha \iff \exists x.\neg\alpha$
  - $\exists x.\alpha \iff \neg\forall x.\neg\alpha$
  - $\neg\exists x.\alpha \iff \forall x.\neg\alpha$
- Examples
  - $\forall x.Likes(x, Icecream)$ equivalent to $\neg\exists x.\neg Likes(x, Icecream)$
  - $\exists x.Likes(x, Broccoli)$ equivalent to $\neg\forall x.\neg Likes(x, Broccoli)$
- Negated restricted quantifiers switch "$\rightarrow$" with "$\wedge$"
  - $\forall x.(P(x) \rightarrow \alpha) \iff \neg\exists x.(P(x) \wedge \neg\alpha)$
  - $\neg\forall x.(P(x) \rightarrow \alpha) \iff \exists x.(P(x) \wedge \neg\alpha)$
  - ...
- Ex: "not all kings are evil" same as "some king is not evil"
  - $\neg\forall x.(King(x) \rightarrow Evil(x)) \iff \exists x.(King(x) \wedge \neg Evil(x))$
- Unsurprising, since $\langle\forall, \exists\rangle$ are $\langle\wedge, \vee\rangle$ over infinite instantiations

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 $\alpha \land \neg\beta$ 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 $\alpha \land \neg\beta$ 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!

Basic Definitions and Properties: 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?

Basic Definitions and Properties: 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?

Basic Definitions and Properties: 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?

Basic Definitions and Properties: 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?

Basic Definitions and Properties: 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?

Basic Definitions and Properties: 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?

Basic Definitions and Properties: Examples

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

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

Exercises

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

Enumeration of Models?

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

  For each number of universe elements n from 1 to ∞
      For each k-ary predicate P_k in the sentence
          For each possible k-ary relation on n objects
              For each constant symbol C in the sentence
                  For each one of n objects C is mapped to
                  ...

- $\Longrightarrow$ 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

Outline

Outline

Term/Subformula Substitutions

Notation

- Substitution: "Subst($\{e_1/e_2\}, e$)" or "$e\{e_1/e_2\}$":
  the expression (term or formula) obtained by substituting every occurrence of e_1 with e_2 in e
  - e_1, e_2 either both terms (term substitution) or both subformulas (subformula substitution)
  - e is either a term or a formula (only term for term substitution)
- Examples:
  - (t. sub.): $(y + 1 = 1 + y)\{y/S(x)\} \Longrightarrow (S(x) + 1 = 1 + S(x))$
  - (s.f. sub.): $(Even(x) \vee Odd(x))\{Even(x)/Odd(S(x))\} \Longrightarrow ((Odd(S(x)) \vee Odd(x))$
- Multiple substitution: $e\{e_1/e_2, e_3/e_4\} \stackrel{\text{def}}{=} (e\{e_1/e_2\})\{e_3/e_4\}$
  - ex: $(P(x, y) \rightarrow Q(x, y))\{x/1, y/2\} \Longrightarrow (P(1, 2) \rightarrow Q(1, 2))$
- If θ is a substitution list and e an expression (formula/term), then we denote the result of a substitution as $e\theta$
  - $e\emptyset = e$
  - $e(\theta_1\theta_2) = (e\theta_1)\theta_2$, denoted as $e\theta_1\theta_2$

Term/Subformula Substitutions

Notation

- Substitution: "Subst($\{e_1/e_2\}, e$)" or "$e\{e_1/e_2\}$":
  the expression (term or formula) obtained by substituting every occurrence of e_1 with e_2 in e
  - e_1, e_2 either both terms (term substitution)
    or both subformulas (subformula substitution)
  - e is either a term or a formula (only term for term substitution)
- Examples:
  - (t. sub.): $(y+1=1+y)\{y/S(x)\} \Longrightarrow (S(x)+1=1+S(x))$
  - (s.f. sub.): $(Even(x) \vee Odd(x))\{Even(x)/Odd(S(x))\} \Longrightarrow$
    $((Odd(S(x)) \vee Odd(x))$
- Multiple substitution: $e\{e_1/e_2, e_3/e_4\} \stackrel{\text{def}}{=} (e\{e_1/e_2\})\{e_3/e_4\}$
  - ex: $(P(x,y) \rightarrow Q(x,y))\{x/1, y/2\} \Longrightarrow (P(1,2) \rightarrow Q(1,2))$
- If θ is a substitution list and e an expression (formula/term), then we denote the result of a substitution as $e\theta$
  - $e\emptyset = e$
  - $e(\theta_1\theta_2) = (e\theta_1)\theta_2$, denoted as $e\theta_1\theta_2$

Term/Subformula Substitutions

Notation

- Substitution: "Subst($\{e_1/e_2\}, e$)" or "$e\{e_1/e_2\}$":
  the expression (term or formula) obtained by substituting every occurrence of e_1 with e_2 in e
  - e_1, e_2 either both terms (term substitution) or both subformulas (subformula substitution)
  - e is either a term or a formula (only term for term substitution)
- Examples:
  - (t. sub.): $(y + 1 = 1 + y)\{y/S(x)\} \Longrightarrow (S(x) + 1 = 1 + S(x))$
  - (s.f. sub.): $(Even(x) \vee Odd(x))\{Even(x)/Odd(S(x))\} \Longrightarrow ((Odd(S(x)) \vee Odd(x))$
- Multiple substitution: $e\{e_1/e_2, e_3/e_4\} \stackrel{\text{def}}{=} (e\{e_1/e_2\})\{e_3/e_4\}$
  - ex: $(P(x, y) \rightarrow Q(x, y))\{x/1, y/2\} \Longrightarrow (P(1, 2) \rightarrow Q(1, 2))$
- If θ is a substitution list and e an expression (formula/term), then we denote the result of a substitution as $e\theta$
  - $e\emptyset = e$
  - $e(\theta_1\theta_2) = (e\theta_1)\theta_2$, denoted as $e\theta_1\theta_2$

Term/Subformula Substitutions

Notation

- Substitution: "Subst($\{e_1/e_2\}, e$)" or "$e\{e_1/e_2\}$":
  the expression (term or formula) obtained by substituting every occurrence of e_1 with e_2 in e
  - e_1, e_2 either both terms (term substitution) or both subformulas (subformula substitution)
  - e is either a term or a formula (only term for term substitution)
- Examples:
  - (t. sub.): $(y+1=1+y)\{y/S(x)\} \Longrightarrow (S(x)+1=1+S(x))$
  - (s.f. sub.): $(Even(x) \vee Odd(x))\{Even(x)/Odd(S(x))\} \Longrightarrow ((Odd(S(x)) \vee Odd(x))$
- Multiple substitution: $e\{e_1/e_2, e_3/e_4\} \stackrel{\text{def}}{=} (e\{e_1/e_2\})\{e_3/e_4\}$
  - ex: $(P(x,y) \rightarrow Q(x,y))\{x/1, y/2\} \Longrightarrow (P(1,2) \rightarrow Q(1,2))$
- If θ is a substitution list and e an expression (formula/term), then we denote the result of a substitution as $e\theta$
  - $e\emptyset = e$
  - $e(\theta_1\theta_2) = (e\theta_1)\theta_2$, denoted as $e\theta_1\theta_2$

Substitution with equivalent terms

Equal-term substitution rule

$$\frac{\Gamma \wedge (t_1 = t_2) \wedge \alpha}{\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}}$$

- Ex: $(S(x) = x + 1) \wedge (0 \neq S(x)) \implies (S(x) = x + 1) \wedge (0 \neq S(x)) \wedge (0 \neq x + 1)$
- Preserves validity: $M(\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}) = M(\Gamma \wedge (t_1 = t_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent terms

Equal-term substitution rule

$$\frac{\Gamma \wedge (t_1 = t_2) \wedge \alpha}{\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}}$$

- Ex: $(S(x) = x + 1) \wedge (0 \neq S(x)) \Longrightarrow$ $(S(x) = x + 1) \wedge (0 \neq S(x)) \wedge (0 \neq x + 1)$
- Preserves validity: $M(\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}) = M(\Gamma \wedge (t_1 = t_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent terms

Equal-term substitution rule

$$\frac{\Gamma \wedge (t_1 = t_2) \wedge \alpha}{\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}}$$

- Ex: $(S(x) = x + 1) \wedge (0 \neq S(x)) \Longrightarrow$
  $(S(x) = x + 1) \wedge (0 \neq S(x)) \wedge (0 \neq x + 1)$
- Preserves validity:
  $M(\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}) = M(\Gamma \wedge (t_1 = t_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent terms

Equal-term substitution rule

$$\frac{\Gamma \wedge (t_1 = t_2) \wedge \alpha}{\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}}$$

- Ex: $(S(x) = x + 1) \wedge (0 \neq S(x)) \Longrightarrow$ $(S(x) = x + 1) \wedge (0 \neq S(x)) \wedge (0 \neq x + 1)$
- Preserves validity: $M(\Gamma \wedge (t_1 = t_2) \wedge \alpha \wedge \alpha\{t_1/t_2\}) = M(\Gamma \wedge (t_1 = t_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent formulas

Equivalent-subformula substitution rule

$$\frac{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha}{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}}$$

- Ex: $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \Longrightarrow (Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \wedge (Odd(S(x)) \vee Odd(x))$
- Preserves validity: $M(\Gamma \wedge (\beta_1 = \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}) = M(\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent formulas

Equivalent-subformula substitution rule

$$\frac{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha}{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}}$$

- Ex: $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \Longrightarrow$ $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \wedge (Odd(S(x)) \vee Odd(x))$
- Preserves validity: $M(\Gamma \wedge (\beta_1 = \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}) = M(\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent formulas

Equivalent-subformula substitution rule

$$\frac{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha}{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}}$$

- Ex: $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \Longrightarrow$ $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \wedge (Odd(S(x)) \vee Odd(x))$
- Preserves validity: $M(\Gamma \wedge (\beta_1 = \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}) = M(\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha)$
- α can be safely dropped from the result

Substitution with equivalent formulas

Equivalent-subformula substitution rule

$$\frac{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha}{\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}}$$

- Ex: $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \Longrightarrow$ $(Even(x) \leftrightarrow Odd(S(x))) \wedge (Even(x) \vee Odd(x)) \wedge (Odd(S(x)) \vee Odd(x))$
- Preserves validity:
  $M(\Gamma \wedge (\beta_1 = \beta_2) \wedge \alpha \wedge \alpha\{\beta_1/\beta_2\}) = M(\Gamma \wedge (\beta_1 \leftrightarrow \beta_2) \wedge \alpha)$
- α can be safely dropped from the result

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:

$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$

  for every variable x and term t
- Ex: $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity: $M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) = M(\Gamma \wedge \forall x.\alpha)$

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:

$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$

  for every variable x and term t
- Ex: $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity:
  $M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) = M(\Gamma \wedge \forall x.\alpha)$

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:

$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$

  for every variable x and term t
- Ex: $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity:
  $M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) = M(\Gamma \wedge \forall x.\alpha)$

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:

$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$

  for every variable x and term t
- Ex: $\forall (x.(King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity:
  $M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) = M(\Gamma \wedge \forall x.\alpha)$

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:
$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$
for every variable x and term t
- Ex: $\forall(x.(King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity:
$M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) = M(\Gamma \wedge \forall x.\alpha)$

Universal Instantiation (UI)

- Every instantiation of a universally quantified-sentence is entailed by it:

$$\frac{\Gamma \wedge \forall x.\alpha}{\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}}$$

  for every variable x and term t
- Ex: $\forall(x.(King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  - $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  - $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  - $(King(Father(John)) \wedge Greedy(Father(John))) \rightarrow Evil(Father(John))$
  - $(King(Father(Father(John))) \wedge Greedy(Father(Father(John)))) \rightarrow Evil(Father(Father(John)))$
  - ...
- Preserves validity:
  $M(\Gamma \wedge \forall x.\alpha \wedge \alpha\{x/t\}) \;=\; M(\Gamma \wedge \forall x.\alpha)$

Existential Instantiation (EI)

- An existentially quantified-sentence can be substituted by one of its instantiation with a fresh constant:
$$\frac{\Gamma \wedge \exists x.\alpha}{\Gamma \wedge \alpha\{x/C\}}$$
for every variable x and for a "fresh" constant C, i.e. a constant which does not appear in $\Gamma \wedge \exists x.\alpha$
- C is a Skolem constant, EI subcase of Skolemization (see later)
- Intuition: if there is an object satisfying some condition, then we give a (new) name to such object
- Ex: $\exists x.(Crown(x) \wedge OnHead(x, John))$
  - $(Crown(C) \wedge OnHead(C, John))$
  - given "There is a crown on John's head", I call "C" such crown
- Preserves satisfiability (aka preserves inferential equivalence)
$M(\Gamma \wedge \alpha\{x/C\}) \neq \emptyset$ *iff* $M(\Gamma \wedge \exists x.\alpha) \neq \emptyset$
(i.e.. $(\Gamma \wedge \alpha\{x/C\}) \models \beta$ iff $(\Gamma \wedge \exists x.\alpha) \models \beta$, for every β)
- Ex from math: $\exists x.(\frac{d(x^y)}{dy} = x^y)$, we call it "e" $\Longrightarrow (\frac{d(e^y)}{dy} = e^y)$

Existential Instantiation (EI)

- An existentially quantified-sentence can be substituted by one of its instantiation with a fresh constant:

$$\frac{\Gamma \wedge \exists x.\alpha}{\Gamma \wedge \alpha\{x/C\}}$$

  for every variable x and for a "fresh" constant C, i.e. a constant which does not appear in $\Gamma \wedge \exists x.\alpha$
- C is a Skolem constant, EI subcase of Skolemization (see later)
- Intuition: if there is an object satisfying some condition, then we give a (new) name to such object
- Ex: $\exists x.(Crown(x) \wedge OnHead(x, John))$
  - $(Crown(C) \wedge OnHead(C, John))$
  - given "There is a crown on John's head", I call "C" such crown
- Preserves satisfiability (aka preserves inferential equivalence)
  $M(\Gamma \wedge \alpha\{x/C\}) \neq \emptyset$ *iff* $M(\Gamma \wedge \exists x.\alpha) \neq \emptyset$
  (i.e.. $(\Gamma \wedge \alpha\{x/C\}) \models \beta$ iff $(\Gamma \wedge \exists x.\alpha) \models \beta$, for every β)
- Ex from math: $\exists x.(\frac{d(x^y)}{dy} = x^y)$, we call it "e" $\Longrightarrow (\frac{d(e^y)}{dy} = e^y)$

Existential Instantiation (EI)

- An existentially quantified-sentence can be substituted by one of its instantiation with a fresh constant:

$$\frac{\Gamma \wedge \exists x.\alpha}{\Gamma \wedge \alpha\{x/C\}}$$

  for every variable x and for a "fresh" constant C, i.e. a constant which does not appear in $\Gamma \wedge \exists x.\alpha$
- C is a Skolem constant, EI subcase of Skolemization (see later)
- Intuition: if there is an object satisfying some condition, then we give a (new) name to such object
- Ex: $\exists x.(Crown(x) \wedge OnHead(x, John))$
  - $(Crown(C) \wedge OnHead(C, John))$
  - given "There is a crown on John's head", I call "C" such crown
- Preserves satisfiability (aka preserves inferential equivalence)
  $M(\Gamma \wedge \alpha\{x/C\}) \neq \emptyset$ *iff* $M(\Gamma \wedge \exists x.\alpha) \neq \emptyset$
  (i.e.. $(\Gamma \wedge \alpha\{x/C\}) \models \beta$ iff $(\Gamma \wedge \exists x.\alpha) \models \beta$, for every β)
- Ex from math: $\exists x.(\frac{d(x^y)}{dy} = x^y)$, we call it "e" $\Longrightarrow (\frac{d(e^y)}{dy} = e^y)$

Existential Instantiation (EI)

- An existentially quantified-sentence can be substituted by one of its instantiation with a fresh constant:
$$\frac{\Gamma \wedge \exists x.\alpha}{\Gamma \wedge \alpha\{x/C\}}$$
for every variable x and for a "fresh" constant C, i.e. a constant which does not appear in $\Gamma \wedge \exists x.\alpha$
- C is a Skolem constant, EI subcase of Skolemization (see later)
- Intuition: if there is an object satisfying some condition, then we give a (new) name to such object
- Ex: $\exists x.(Crown(x) \wedge OnHead(x, John))$
  - $(Crown(C) \wedge OnHead(C, John))$
  - given "There is a crown on John's head", I call "C" such crown
- Preserves satisfiability (aka preserves inferential equivalence)
$M(\Gamma \wedge \alpha\{x/C\}) \neq \emptyset$ *iff* $M(\Gamma \wedge \exists x.\alpha) \neq \emptyset$
(i.e.. $(\Gamma \wedge \alpha\{x/C\}) \models \beta$ iff $(\Gamma \wedge \exists x.\alpha) \models \beta$, for every β)
- Ex from math: $\exists x.(\frac{d(x^y)}{dy} = x^y)$, we call it "e" $\Longrightarrow (\frac{d(e^y)}{dy} = e^y)$

Existential Instantiation (EI)

- An existentially quantified-sentence can be substituted by one of its instantiation with a fresh constant:
$$\frac{\Gamma \wedge \exists x.\alpha}{\Gamma \wedge \alpha\{x/C\}}$$
for every variable x and for a "fresh" constant C, i.e. a constant which does not appear in $\Gamma \wedge \exists x.\alpha$
- C is a Skolem constant, EI subcase of Skolemization (see later)
- Intuition: if there is an object satisfying some condition, then we give a (new) name to such object
- Ex: $\exists x.(Crown(x) \wedge OnHead(x, John))$
  - $(Crown(C) \wedge OnHead(C, John))$
  - given "There is a crown on John's head", I call "C" such crown
- Preserves satisfiability (aka preserves inferential equivalence)
$M(\Gamma \wedge \alpha\{x/C\}) \neq \emptyset$ *iff* $M(\Gamma \wedge \exists x.\alpha) \neq \emptyset$
(i.e.. $(\Gamma \wedge \alpha\{x/C\}) \models \beta$ iff $(\Gamma \wedge \exists x.\alpha) \models \beta$, for every β)
- Ex from math: $\exists x.(\frac{d(x^y)}{dy} = x^y)$, we call it "e" $\Longrightarrow$ $(\frac{d(e^y)}{dy} = e^y)$

Remarks

- About Universal Instantiation:
  - UI can be applied several times to add new sentences;
  - the new Γ is logically equivalent to the old Γ
- About Existential Instantiation:
  - EI can be applied once to replace the existential sentence;
  - the new Γ is not equivalent to the old,
  - but is (un)satisfiable iff the old Γ is (un)satisfiable

  $\Longrightarrow$ the new Γ can infer β iff the old Γ can infer β

Before applying UI or EI, sentences must be rewritten s.t. negations (even when implicit) must be pushed inside the quantifications:

- $\neg\forall x.\alpha \Longrightarrow \exists x.\neg\alpha$
- $\neg\exists x.\alpha \Longrightarrow \forall x.\neg\alpha$
- ex: $(\forall x.P(x) \rightarrow \neg\exists y.Q(y))$

  $\Longrightarrow (\neg\forall x.P(x) \vee \neg\exists y.Q(y))$

  $\Longrightarrow (\exists x.\neg P(x) \vee \forall y.\neg Q(y))$

Remarks

- About Universal Instantiation:
  - UI can be applied several times to add new sentences;
  - the new Γ is logically equivalent to the old Γ
- About Existential Instantiation:
  - EI can be applied once to replace the existential sentence;
  - the new Γ is not equivalent to the old,
  - but is (un)satisfiable iff the old Γ is (un)satisfiable

  $\Longrightarrow$ the new Γ can infer β iff the old Γ can infer β

Before applying UI or EI, sentences must be rewritten s.t. negations (even when implicit) must be pushed inside the quantifications:

- $\neg\forall x.\alpha \Longrightarrow \exists x.\neg\alpha$
- $\neg\exists x.\alpha \Longrightarrow \forall x.\neg\alpha$
- ex: $(\forall x.P(x) \rightarrow \neg\exists y.Q(y))$

  $\Longrightarrow (\neg\forall x.P(x) \vee \neg\exists y.Q(y))$

  $\Longrightarrow (\exists x.\neg P(x) \vee \forall y.\neg Q(y))$

Remarks

- About Universal Instantiation:
  - UI can be applied several times to add new sentences;
  - the new Γ is logically equivalent to the old Γ
- About Existential Instantiation:
  - EI can be applied once to replace the existential sentence;
  - the new Γ is not equivalent to the old,
  - but is (un)satisfiable iff the old Γ is (un)satisfiable

  $\Longrightarrow$ the new Γ can infer β iff the old Γ can infer β

Before applying UI or EI, sentences must be rewritten s.t. negations (even when implicit) must be pushed inside the quantifications:

- $\neg\forall x.\alpha \Longrightarrow \exists x.\neg\alpha$
- $\neg\exists x.\alpha \Longrightarrow \forall x.\neg\alpha$
- ex: $(\forall x.P(x) \rightarrow \neg\exists y.Q(y))$
  $\Longrightarrow (\neg\forall x.P(x) \vee \neg\exists y.Q(y))$
  $\Longrightarrow (\exists x.\neg P(x) \vee \forall y.\neg Q(y))$

Remarks

- About Universal Instantiation:
  - UI can be applied several times to add new sentences;
  - the new Γ is logically equivalent to the old Γ
- About Existential Instantiation:
  - EI can be applied once to replace the existential sentence;
  - the new Γ is not equivalent to the old,
  - but is (un)satisfiable iff the old Γ is (un)satisfiable

  $\Longrightarrow$ the new Γ can infer β iff the old Γ can infer β

Before applying UI or EI, sentences must be rewritten s.t. negations (even when implicit) must be pushed inside the quantifications:

- $\neg\forall x.\alpha \Longrightarrow \exists x.\neg\alpha$
- $\neg\exists x.\alpha \Longrightarrow \forall x.\neg\alpha$
- ex: $(\forall x.P(x) \rightarrow \neg\exists y.Q(y))$
  $\Longrightarrow (\neg\forall x.P(x) \vee \neg\exists y.Q(y))$
  $\Longrightarrow (\exists x.\neg P(x) \vee \forall y.\neg Q(y))$

Outline

Reduction to Propositional Inference

- Idea: Convert $(\Gamma \wedge \neg\alpha)$ to PL (aka propositionalization)
  $\Longrightarrow$ use a PL SAT solver to check PL (un)satisfiability
- Trick:
  - replace variables with ground terms, creating all possible instantiations of quantified sentences
  - convert atomic sentences into propositional symbols

    e.g. "King(John)" $\Longrightarrow$ "King_John",
    e.g. "Brother(John,Richard)" $\Longrightarrow$ "Brother_John-Richard",
- Theorem: (Herbrand, 1930)
  If a ground sentence α is entailed by an FOL Γ,
  then it is entailed by a finite subset of the propositional Γ

  $\Longrightarrow$ A ground sentence is entailed by the propositionalized Γ if it is entailed by original Γ
  $\Longrightarrow$ Every FOL Γ can be propositionalized s.t. to preserve entailment
- The vice-versa does not hold
  $\Longrightarrow$ works if α is entailed, loops if α is not entailed

Reduction to Propositional Inference

- Idea: Convert $(\Gamma \wedge \neg\alpha)$ to PL (aka propositionalization)
  $\Longrightarrow$ use a PL SAT solver to check PL (un)satisfiability
- Trick:
  - replace variables with ground terms, creating all possible instantiations of quantified sentences
  - convert atomic sentences into propositional symbols

    e.g. "King(John)" $\Longrightarrow$ "King_John",
    e.g. "Brother(John,Richard)" $\Longrightarrow$ "Brother_John-Richard",
- Theorem: (Herbrand, 1930)
  If a ground sentence α is entailed by an FOL Γ,
  then it is entailed by a finite subset of the propositional Γ

  $\Longrightarrow$ A ground sentence is entailed by the propositionalized Γ if it is entailed by original Γ

  $\Longrightarrow$ Every FOL Γ can be propositionalized s.t. to preserve entailment
- The vice-versa does not hold
  $\Longrightarrow$ works if α is entailed, loops if α is not entailed

Reduction to Propositional Inference

- Idea: Convert $(\Gamma \land \neg\alpha)$ to PL (aka propositionalization)
  $\Longrightarrow$ use a PL SAT solver to check PL (un)satisfiability
- Trick:
  - replace variables with ground terms, creating all possible instantiations of quantified sentences
  - convert atomic sentences into propositional symbols

    e.g. "King(John)" $\Longrightarrow$ "King_John",
    e.g. "Brother(John,Richard)" $\Longrightarrow$ "Brother_John-Richard",
- Theorem: (Herbrand, 1930)
  If a ground sentence α is entailed by an FOL Γ,
  then it is entailed by a finite subset of the propositional Γ

  $\Longrightarrow$ A ground sentence is entailed by the propositionalized Γ if it is entailed by original Γ

  $\Longrightarrow$ Every FOL Γ can be propositionalized s.t. to preserve entailment
- The vice-versa does not hold
  $\Longrightarrow$ works if α is entailed, loops if α is not entailed

Reduction to Propositional Inference

- Idea: Convert ($\Gamma \wedge \neg\alpha$) to PL (aka propositionalization)
  $\Longrightarrow$ use a PL SAT solver to check PL (un)satisfiability
- Trick:
  - replace variables with ground terms, creating all possible instantiations of quantified sentences
  - convert atomic sentences into propositional symbols

    e.g. "King(John)" $\Longrightarrow$ "King_John",
    e.g. "Brother(John,Richard)" $\Longrightarrow$ "Brother_John-Richard",
- Theorem: (Herbrand, 1930)
  If a ground sentence α is entailed by an FOL Γ,
  then it is entailed by a finite subset of the propositional Γ
  $\Longrightarrow$ A ground sentence is entailed by the propositionalized Γ if it is entailed by original Γ
  $\Longrightarrow$ Every FOL Γ can be propositionalized s.t. to preserve entailment
- The vice-versa does not hold
  $\Longrightarrow$ works if α is entailed, loops if α is not entailed

Reduction to Propositional Inference: Example

- Suppose Γ contains only:

  $\forall x.((King(x) \land Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways:

  $(King(John) \land Greedy(John)) \rightarrow Evil(John)$
  $(King(Richard) \land Greedy(Richard)) \rightarrow Evil(Richard)$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- The new Γ is propositionalized:

  $(King\_John \land Greedy\_John) \rightarrow Evil\_John$
  $(King\_Richard \land Greedy\_Richard) \rightarrow Evil\_Richard$
  $King\_John$
  $Greedy\_John$
  $Brother\_Richard\text{-}John$

- $Evil\_John$ entailed by new Γ ($Evil(John)$ entailed by old Γ)

Reduction to Propositional Inference: Example

- Suppose Γ contains only:

$\forall x.((King(x) \land Greedy(x)) \rightarrow Evil(x))$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways:

$(King(John) \land Greedy(John)) \rightarrow Evil(John)$
$(King(Richard) \land Greedy(Richard)) \rightarrow Evil(Richard)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

- The new Γ is propositionalized:

$(King\_John \land Greedy\_John) \rightarrow Evil\_John$
$(King\_Richard \land Greedy\_Richard) \rightarrow Evil\_Richard$
$King\_John$
$Greedy\_John$
$Brother\_Richard\text{-}John$

- *Evil_John* entailed by new Γ (*Evil*(*John*) entailed by old Γ)

Reduction to Propositional Inference: Example

- Suppose Γ contains only:

  $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways:

  $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- The new Γ is propositionalized:

  $(King\_John \wedge Greedy\_John) \rightarrow Evil\_John$
  $(King\_Richard \wedge Greedy\_Richard) \rightarrow Evil\_Richard$
  $King\_John$
  $Greedy\_John$
  $Brother\_Richard\text{-}John$

- *Evil_John* entailed by new Γ (*Evil*(*John*) entailed by old Γ)

Reduction to Propositional Inference: Example

- Suppose Γ contains only:

  $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- Instantiating the universal sentence in all possible ways:

  $(King(John) \wedge Greedy(John)) \rightarrow Evil(John)$
  $(King(Richard) \wedge Greedy(Richard)) \rightarrow Evil(Richard)$
  $King(John)$
  $Greedy(John)$
  $Brother(Richard, John)$

- The new Γ is propositionalized:

  $(King\_John \wedge Greedy\_John) \rightarrow Evil\_John$
  $(King\_Richard \wedge Greedy\_Richard) \rightarrow Evil\_Richard$
  $King\_John$
  $Greedy\_John$
  $Brother\_Richard\text{-}John$

- $Evil\_John$ entailed by new Γ ($Evil(John)$ entailed by old Γ)

Problems with Propositionalization

- Propositionalization generates lots of irrelevant sentences

  produces irrelevant atoms like Greedy(Richard)

  $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $\forall y.Greedy(y)$
  $Brother(Richard, John)$

$\Longrightarrow$ produces irrelevant atoms like Greedy(Richard)

- With p k-ary predicates and n constants, $p \cdot n^k$ instantiations
- What happens with function symbols?

Problems with Propositionalization

- Propositionalization generates lots of irrelevant sentences

  produces irrelevant atoms like Greedy(Richard)

  $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $\forall y.Greedy(y)$
  $Brother(Richard, John)$

$\Longrightarrow$ produces irrelevant atoms like Greedy(Richard)

- With p k-ary predicates and n constants, $p \cdot n^k$ instantiations
- What happens with function symbols?

Problems with Propositionalization

- Propositionalization generates lots of irrelevant sentences

  produces irrelevant atoms like Greedy(Richard)

  $\forall x.((King(x) \wedge Greedy(x)) \rightarrow Evil(x))$
  $King(John)$
  $\forall y.Greedy(y)$
  $Brother(Richard, John)$

$\Longrightarrow$ produces irrelevant atoms like Greedy(Richard)

- With p k-ary predicates and n constants, $p \cdot n^k$ instantiations
- What happens with function symbols?

Problems with Propositionalization [cont.]

- Problem: nested function applications
  - e.g. Father(John), Father(Father(John)), Father(Father(Father(John))), ...

  $\Longrightarrow$ infinite instantiations
- Actual Trick: for k = 0 to ∞, use terms of function nesting depth k
  - create propositionalized Γ by instantiating depth-k terms
  - if $\Gamma \models \alpha$, then will find a contradiction for some finite k
  - if $\Gamma \not\models \alpha$, may find a loop forever
- Theorem: (Turing, 1936), (Church, 1936):
  Entailment in FOL is semidecidable

Problems with Propositionalization [cont.]

- Problem: nested function applications
  - e.g. Father(John), Father(Father(John)), Father(Father(Father(John))), ...

  $\Longrightarrow$ infinite instantiations
- Actual Trick: for k = 0 to ∞, use terms of function nesting depth k
  - create propositionalized Γ by instantiating depth-k terms
  - if Γ $\models \alpha$, then will find a contradiction for some finite k
  - if Γ $\not\models \alpha$, may find a loop forever
- Theorem: (Turing, 1936), (Church, 1936): Entailment in FOL is semidecidable

Problems with Propositionalization [cont.]

- Problem: nested function applications
  - e.g. Father(John), Father(Father(John)), Father(Father(Father(John))), ...

  $\Longrightarrow$ infinite instantiations
- Actual Trick: for k = 0 to ∞, use terms of function nesting depth k
  - create propositionalized Γ by instantiating depth-k terms
  - if $\Gamma \models \alpha$, then will find a contradiction for some finite k
  - if $\Gamma \not\models \alpha$, may find a loop forever
- Theorem: (Turing, 1936), (Church, 1936): Entailment in FOL is semidecidable

Outline

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:

  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\quad \{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes

$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

- $\{\forall x.\alpha, \forall x.\beta\} \Longleftrightarrow \{\forall x_1.\alpha\{x/x_1\}, \forall x_2.\beta\{x/x_2\}\}$, s.t. x_1, x_2 new

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $\mathit{Unify}(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(\mathit{John}, \mathit{Jane})) = \{x/\mathit{Jane}\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(y, \mathit{OJ})) = \{x/\mathit{OJ}, y/\mathit{John}\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(y, \mathit{Mother}(y))) =$
  $\{y/\mathit{John}, x/\mathit{Mother}(\mathit{John})\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(x, \mathit{OJ})) = \mathit{FAIL} : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$\mathit{Unify}(\mathit{Knows}(\mathit{John}, x_1), \mathit{Knows}(x_2, \mathit{OJ})) = \{x_1/\mathit{OBJ}, x_2/\mathit{John}\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\quad \{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $\mathit{Unify}(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(\mathit{John}, \mathit{Jane})) = \{x/\mathit{Jane}\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(y, \mathit{OJ})) = \{x/\mathit{OJ}, y/\mathit{John}\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(y, \mathit{Mother}(y))) =$
  $\{y/\mathit{John}, x/\mathit{Mother}(\mathit{John})\}$
  $\mathit{Unify}(\mathit{Knows}(\mathit{John}, x), \mathit{Knows}(x, \mathit{OJ})) = \mathit{FAIL} : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$\mathit{Unify}(\mathit{Knows}(\mathit{John}, x_1), \mathit{Knows}(x_2, \mathit{OJ})) = \{x_1/\mathit{OBJ}, x_2/\mathit{John}\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\quad \{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

Unification

- Unification: Given $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes

$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

- $\{\forall x.\alpha, \forall x.\beta\} \Longleftrightarrow \{\forall x_1.\alpha\{x/x_1\}, \forall x_2.\beta\{x/x_2\}\}$, s.t. x_1, x_2 new

Unification

- Unification: Given $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i \theta = \alpha_i \theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

- $\{\forall x.\alpha, \forall x.\beta\} \Longleftrightarrow \{\forall x_1.\alpha\{x/x_1\}, \forall x_2.\beta\{x/x_2\}\}$, s.t. x_1, x_2 new

Unification

- Unification: Given $\langle\alpha'_1,\ \alpha'_2,\ \ldots,\ \alpha'_k\rangle$ and $\langle\alpha_1, \alpha_2, \ldots, \alpha_k\rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle\alpha'_1,\ \alpha'_2,\ \ldots,\ \alpha'_k\rangle$ and $\langle\alpha_1, \alpha_2, \ldots, \alpha_k\rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$

  - $\{\forall x.\alpha, \forall x.\beta\} \Longleftrightarrow \{\forall x_1.\alpha\{x/x_1\}, \forall x_2.\beta\{x/x_2\}\}$, s.t. x_1, x_2 new

Unification

- Unification: Given $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$, find a variable substitution θ s.t. θ s.t. $\alpha'_i\theta = \alpha_i\theta$, for all $i \in 1..k$
  - θ is called a unifier for $\langle \alpha'_1, \ \alpha'_2, \ ..., \ \alpha'_k \rangle$ and $\langle \alpha_1, \alpha_2, ..., \alpha_k \rangle$
  - $Unify(\alpha, \beta) = \theta$ iff $\alpha\theta = \beta\theta$
- Ex:
  $Unify(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
  $Unify(Knows(John, x), Knows(y, OJ)) = \{x/OJ, y/John\}$
  $Unify(Knows(John, x), Knows(y, Mother(y))) =$
  $\{y/John, x/Mother(John)\}$
  $Unify(Knows(John, x), Knows(x, OJ)) = FAIL : x/?$
- Different (implicitly-universally-quantified) formulas should use different variables

$\Longrightarrow$ (Standardizing apart): rename variables to avoid name clashes
$Unify(Knows(John, x_1), Knows(x_2, OJ)) = \{x_1/OBJ, x_2/John\}$
  - $\{\forall x.\alpha, \forall x.\beta\} \Longleftrightarrow \{\forall x_1.\alpha\{x/x_1\}, \forall x_2.\beta\{x/x_2\}\}$, s.t. x_1, x_2 new

Most-General Unifier (MGU)

- Unifiers are not unique
  - ex: $Unify(Knows(John, x), Knows(y, z))$
    could return $\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$
- Given α, β, the unifier θ_1 is more general than the unifier θ_2 for α, β if exists θ_3 s.t. $\theta_2 = \theta_1\theta_3$
  - ex: $\{y/John, x/z\}$ more general than $\{y/John, x/John, z/John\}$:
    $\{y/John, x/John, z/John\} = \{y/John, x/z\}\{z/John\}$
- Theorem: If exists an unifier for α, β, then exists a most general unifier (MGU) θ for α, β
  - Ex: $\{y/John, x/z\}$ MGU for $Knows(John, x), Knows(y, z)$
  - Ex: an MGU is unique modulo variable renaming
- UNIFY() returns the MGU between two (lists of) formulas
  - efficiency optimizations based on predicate/term indexing techniques (see AIMA if interested)

Most-General Unifier (MGU)

- Unifiers are not unique
  - ex: $Unify(Knows(John, x), Knows(y, z))$ could return $\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$
- Given α, β, the unifier θ_1 is more general than the unifier θ_2 for α, β if exists θ_3 s.t. $\theta_2 = \theta_1\theta_3$
  - ex: $\{y/John, x/z\}$ more general than $\{y/John, x/John, z/John\}$: $\{y/John, x/John, z/John\} = \{y/John, x/z\}\{z/John\}$
- Theorem: If exists an unifier for α, β, then exists a most general unifier (MGU) θ for α, β
  - Ex: $\{y/John, x/z\}$ MGU for $Knows(John, x), Knows(y, z)$
  - Ex: an MGU is unique modulo variable renaming
- UNIFY() returns the MGU between two (lists of) formulas
  - efficiency optimizations based on predicate/term indexing techniques (see AIMA if interested)

Most-General Unifier (MGU)

- Unifiers are not unique
  - ex: $Unify(Knows(John, x), Knows(y, z))$ could return $\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$
- Given α, β, the unifier θ_1 is more general than the unifier θ_2 for α, β if exists θ_3 s.t. $\theta_2 = \theta_1\theta_3$
  - ex: $\{y/John, x/z\}$ more general than $\{y/John, x/John, z/John\}$: $\{y/John, x/John, z/John\} = \{y/John, x/z\}\{z/John\}$
- Theorem: If exists an unifier for α, β, then exists a most general unifier (MGU) θ for α, β
  - Ex: $\{y/John, x/z\}$ MGU for $Knows(John, x), Knows(y, z)$
  - Ex: an MGU is unique modulo variable renaming
- UNIFY() returns the MGU between two (lists of) formulas
  - efficiency optimizations based on predicate/term indexing techniques (see AIMA if interested)

Most-General Unifier (MGU)

- Unifiers are not unique
  - ex: $Unify(Knows(John, x), Knows(y, z))$
    could return $\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$
- Given α, β, the unifier θ_1 is more general than the unifier θ_2 for α, β if exists θ_3 s.t. $\theta_2 = \theta_1\theta_3$
  - ex: $\{y/John, x/z\}$ more general than $\{y/John, x/John, z/John\}$:
    $\{y/John, x/John, z/John\} = \{y/John, x/z\}\{z/John\}$
- Theorem: If exists an unifier for α, β, then exists a most general unifier (MGU) θ for α, β
  - Ex: $\{y/John, x/z\}$ MGU for $Knows(John, x), Knows(y, z)$
  - Ex: an MGU is unique modulo variable renaming
- UNIFY() returns the MGU between two (lists of) formulas
  - efficiency optimizations based on predicate/term indexing techniques (see AIMA if interested)

The Procedure Unify

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound expression
    y, a variable, constant, list, or compound expression
    θ, the substitution built up so far (optional, defaults to empty)

  if θ = failure **then return** failure
  else if $x = y$ **then return** θ
  else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) **and** COMPOUND?(y) **then**
    return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) **and** LIST?(y) **then**
    return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure

---

function UNIFY-VAR(var, x, θ) **returns** a substitution

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Outline

Outline

Conjunctive Normal Form (CNF)

- A FOL formula φ is in Conjunctive normal form iff it is a conjunction of disjunctions of quantifier-free literal:

$$\bigwedge_{i=1}^{L} \bigvee_{j_i=1}^{K_i} l_{j_i}$$

  - the disjunctions of literals $\bigvee_{j_i=1}^{K_i} l_{j_i}$ are called clauses
  - every literal a quantifier-free atom or its negation
  - free variables implicitly universally quantified
- Easier to handle: list of lists of literals.
  $\Longrightarrow$ no reasoning on the recursive structure of the formula
- Ex: $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

FOL CNF Conversion $CNF(\varphi)$

Convert into NNF

Every FOL formula φ can be reduced into CNF:

1. Eliminate implications and biconditionals:

$$\begin{array}{lcl} \alpha \rightarrow \beta & \Longrightarrow & \neg\alpha \vee \beta \\ \alpha \leftrightarrow \beta & \Longrightarrow & (\neg\alpha \vee \beta) \wedge (\alpha \vee \neg\beta) \end{array}$$

2. Push inwards negations recursively:

$$\begin{array}{lcl} \neg(\alpha \wedge \beta) & \Longrightarrow & \neg\alpha \vee \neg\beta \\ \neg(\alpha \vee \beta) & \Longrightarrow & \neg\alpha \wedge \neg\beta \\ \neg\neg\alpha & \Longrightarrow & \alpha \\ \neg\forall x.\alpha & \Longrightarrow & \exists x.\neg\alpha \\ \neg\exists x.\alpha & \Longrightarrow & \forall x.\neg\alpha \end{array}$$

$\Longrightarrow$ Negation normal form: negations only in front of atomic formulae

$\Longrightarrow$ quantified subformulas occur only with positive polarity

FOL CNF Conversion $CNF(\varphi)$

Convert into NNF

Every FOL formula φ can be reduced into CNF:

1. Eliminate implications and biconditionals:

$$\begin{array}{lcl} \alpha \rightarrow \beta & \Longrightarrow & \neg\alpha \vee \beta \\ \alpha \leftrightarrow \beta & \Longrightarrow & (\neg\alpha \vee \beta) \wedge (\alpha \vee \neg\beta) \end{array}$$

2. Push inwards negations recursively:

$$\begin{array}{lcl} \neg(\alpha \wedge \beta) & \Longrightarrow & \neg\alpha \vee \neg\beta \\ \neg(\alpha \vee \beta) & \Longrightarrow & \neg\alpha \wedge \neg\beta \\ \neg\neg\alpha & \Longrightarrow & \alpha \\ \neg\forall x.\alpha & \Longrightarrow & \exists x.\neg\alpha \\ \neg\exists x.\alpha & \Longrightarrow & \forall x.\neg\alpha \end{array}$$

$\Longrightarrow$ Negation normal form: negations only in front of atomic formulae

$\Longrightarrow$ quantified subformulas occur only with positive polarity

FOL CNF Conversion $CNF(\varphi)$ [cont.]

Remove quantifiers

3. Standardize variables: each quantifier should use a different var
   $(\forall x.\exists y.\alpha) \land \exists y.\beta \land \forall x.\gamma \implies (\forall x.\exists y.\alpha) \land \exists y_1.\beta\{y/y_1\} \land \forall x_1.\gamma\{x/x_1\}$
4. Skolemize (a generalization of EI):
   Each existential variable is replaced by a fresh Skolem function applied to the enclosing universally-quantified variables
   $$\begin{array}{lcl} \exists y.\alpha & \implies & \alpha\{y/c\} \\ \forall x.(...\exists y.\alpha...) & \implies & \forall x.(...\alpha\{y/F_1(x)\}...) \\ \forall x_1 x_2.(...\exists y.\alpha...) & \implies & \forall x_1 x_2.(...\alpha\{y/F_1(x_1,x_2)...)\} \\ \exists y_1 \forall x_1 x_2 \exists y_2 \forall x_3 \exists y_3.\alpha & \implies & \forall x_1 x_2 x_3. \\ & & \quad \alpha\{y_1/c, y_2/F_1(x_1,x_2), y_3/F_2(x_1,x_2,x_3)\} \end{array}$$
   Ex: $\forall x \exists y.Father(x,y) \implies \forall x.Father(x,s(x))$
   ($s(x)$ implictly means "son of x" although s() is a fresh function)
5. Drop universal quantifiers: $\forall x_1 ... x_k.\alpha \implies \alpha$
   $\implies$ free variables implicitly universally quantified

FOL CNF Conversion $CNF(\varphi)$ [cont.]

Remove quantifiers

3. **Standardize variables**: each quantifier should use a different var
   $(\forall x.\exists y.\alpha) \wedge \exists y.\beta \wedge \forall x.\gamma \implies (\forall x.\exists y.\alpha) \wedge \exists y_1.\beta\{y/y_1\} \wedge \forall x_1.\gamma\{x/x_1\}$

4. **Skolemize** (a generalization of EI):
   Each existential variable is replaced by a fresh Skolem function applied to the enclosing universally-quantified variables

$$
\begin{array}{lcl}
\exists y.\alpha & \implies & \alpha\{y/c\} \\
\forall x.(...\exists y.\alpha...) & \implies & \forall x.(...\alpha\{y/F_1(x)\}...) \\
\forall x_1 x_2.(...\exists y.\alpha...) & \implies & \forall x_1 x_2.(...\alpha\{y/F_1(x_1,x_2)...)\} \\
\exists y_1 \forall x_1 x_2 \exists y_2 \forall x_3 \exists y_3.\alpha & \implies & \forall x_1 x_2 x_3. \\
 & & \quad \alpha\{y_1/c, y_2/F_1(x_1,x_2), y_3/F_2(x_1,x_2,x_3)\}
\end{array}
$$

   Ex: $\forall x \exists y.Father(x,y) \implies \forall x.Father(x, s(x))$
   ($s(x)$ implictly means "son of x" although s() is a fresh function)

5. **Drop universal quantifiers:** $\forall x_1 ... x_k.\alpha \implies \alpha$
   $\implies$ free variables implicitly universally quantified

FOL CNF Conversion $CNF(\varphi)$ [cont.]

Remove quantifiers

3. Standardize variables: each quantifier should use a different var
   $(\forall x.\exists y.\alpha) \wedge \exists y.\beta \wedge \forall x.\gamma \implies (\forall x.\exists y.\alpha) \wedge \exists y_1.\beta\{y/y_1\} \wedge \forall x_1.\gamma\{x/x_1\}$

4. Skolemize (a generalization of EI):
   Each existential variable is replaced by a fresh Skolem function applied to the enclosing universally-quantified variables

$$
\begin{array}{lll}
\exists y.\alpha & \implies & \alpha\{y/c\} \\
\forall x.(...\exists y.\alpha...) & \implies & \forall x.(...\alpha\{y/F_1(x)\}...) \\
\forall x_1 x_2.(...\exists y.\alpha...) & \implies & \forall x_1 x_2.(...\alpha\{y/F_1(x_1,x_2)...)\} \\
\exists y_1 \forall x_1 x_2 \exists y_2 \forall x_3 \exists y_3.\alpha & \implies & \forall x_1 x_2 x_3. \\
 & & \quad \alpha\{y_1/c, y_2/F_1(x_1,x_2), y_3/F_2(x_1,x_2,x_3)\}
\end{array}
$$

   Ex: $\forall x \exists y.Father(x,y) \implies \forall x.Father(x,s(x))$
   ($s(x)$ implictly means "son of x" although s() is a fresh function)

5. Drop universal quantifiers: $\forall x_1 ... x_k.\alpha \implies \alpha$
   $\implies$ free variables implicitly universally quantified

FOL CNF Conversion $CNF(\varphi)$ [cont.]

Remove quantifiers

3. Standardize variables: each quantifier should use a different var
   $(\forall x.\exists y.\alpha) \land \exists y.\beta \land \forall x.\gamma \implies (\forall x.\exists y.\alpha) \land \exists y_1.\beta\{y/y_1\} \land \forall x_1.\gamma\{x/x_1\}$
4. Skolemize (a generalization of EI):
   Each existential variable is replaced by a fresh Skolem function applied to the enclosing universally-quantified variables
   $$\begin{array}{lcl} \exists y.\alpha & \implies & \alpha\{y/c\} \\ \forall x.(...\exists y.\alpha...) & \implies & \forall x.(...\alpha\{y/F_1(x)\}...) \\ \forall x_1 x_2.(...\exists y.\alpha...) & \implies & \forall x_1 x_2.(...\alpha\{y/F_1(x_1, x_2)...)\} \\ \exists y_1 \forall x_1 x_2 \exists y_2 \forall x_3 \exists y_3.\alpha & \implies & \forall x_1 x_2 x_3. \\ & & \quad \alpha\{y_1/c, y_2/F_1(x_1, x_2), y_3/F_2(x_1, x_2, x_3)\} \end{array}$$
   Ex: $\forall x \exists y.Father(x, y) \implies \forall x.Father(x, s(x))$
   ($s(x)$ implictly means "son of x" although s() is a fresh function)
5. Drop universal quantifiers: $\forall x_1 ... x_k.\alpha \implies \alpha$
   $\implies$ free variables implicitly universally quantified

CNF-ize propositionally

6. CNF-ize propositionally (see previous chapters):
   either apply recursively the DeMorgan's Rule:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
   or rename subformulas and add definitions:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (B \vee \gamma) \wedge CNF(B \leftrightarrow (\alpha \wedge \beta))$

Preserves satisfiability: $M(\varphi) \neq \emptyset$ iff $M(CNF(\varphi)) \neq \emptyset$

FOL CNF Conversion $CNF(\varphi)$ [cont.]

CNF-ize propositionally

6. CNF-ize propositionally (see previous chapters):
   either apply recursively the DeMorgan's Rule:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
   or rename subformulas and add definitions:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (B \vee \gamma) \wedge CNF(B \leftrightarrow (\alpha \wedge \beta))$

Preserves satisfiability: $M(\varphi) \neq \emptyset$ iff $M(CNF(\varphi)) \neq \emptyset$

FOL CNF Conversion $CNF(\varphi)$ [cont.]

CNF-ize propositionally

6. CNF-ize propositionally (see previous chapters):
   either apply recursively the DeMorgan's Rule:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$
   or rename subformulas and add definitions:
   $(\alpha \wedge \beta) \vee \gamma \quad \Longrightarrow \quad (B \vee \gamma) \wedge CNF(B \leftrightarrow (\alpha \wedge \beta))$

Preserves satisfiability: $M(\varphi) \neq \emptyset$ iff $M(CNF(\varphi)) \neq \emptyset$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:

   $\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$

2. Push inwards negations recursively:

   $\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$

3. Standardize variables:

   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z.Loves(z,x)])$

4. Skolemize:

   $\forall x.([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)])$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")

5. Drop universal quantifiers::

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

6. CNF-ize propositionally:

   $(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [\exists y.Loves(y, x)])$

1. Eliminate implications and biconditionals:
   $$\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y.Loves(y, x)])$$
2. Push inwards negations recursively:
   $$\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y.Loves(y, x)])$$
   $$\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x, y))] \vee [\exists y.Loves(y, x)])$$
   $$\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x, y))] \vee [\exists y.Loves(y, x)])$$
3. Standardize variables:
   $$\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x, y))] \vee [\exists z.Loves(z, x)])$$
4. Skolemize:
   $$\forall x.([Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee [Loves(G(x), x)])$$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")
5. Drop universal quantifiers::
   $$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee [Loves(G(x), x)]$$
6. CNF-ize propositionally:
   $$(Animal(F(x)) \vee Loves(G(x), x)) \wedge (\neg Loves(x, F(x)) \vee Loves(G(x), x))$$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"

$\forall x.([\forall y.(Animal(y) \to Loves(x,y))] \to [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:

   $\forall x.(\neg[\forall y.(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y.Loves(y,x)])$

2. Push inwards negations recursively:

   $\forall x.([\exists y.\neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(\neg\neg Animal(y) \land \neg Loves(x,y))] \lor [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(Animal(y) \land \neg Loves(x,y))] \lor [\exists y.Loves(y,x)])$

3. Standardize variables:

   $\forall x.([\exists y.(Animal(y) \land \neg Loves(x,y))] \lor [\exists z.Loves(z,x)])$

4. Skolemize:

   $\forall x.([Animal(F(x)) \land \neg Loves(x,F(x))] \lor [Loves(G(x),x)])$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")

5. Drop universal quantifiers::

   $[Animal(F(x)) \land \neg Loves(x,F(x))] \lor [Loves(G(x),x)]$

6. CNF-ize propositionally:

   $(Animal(F(x)) \lor Loves(G(x),x)) \land (\neg Loves(x,F(x)) \lor Loves(G(x),x))$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:
   $$\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$$
2. Push inwards negations recursively:
   $$\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$$
   $$\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$$
   $$\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$$
3. Standardize variables:
   $$\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z.Loves(z,x)])$$
4. Skolemize:
   $$\forall x.([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)])$$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")
5. Drop universal quantifiers::
   $$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$$
6. CNF-ize propositionally:
   $$(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:

   $\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$

2. Push inwards negations recursively:

   $\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$

3. Standardize variables:

   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z.Loves(z,x)])$

4. Skolemize:

   $\forall x.([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)])$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")

5. Drop universal quantifiers::

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

6. CNF-ize propositionally:

   $(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

Conversion to CNF: Example

Consider: “Everyone who loves all animals is loved by someone”
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:
   $\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$
2. Push inwards negations recursively:
   $\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$
3. Standardize variables:
   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z.Loves(z,x)])$
4. Skolemize:
   $\forall x.([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)])$
   (F(x): “an animal unloved by x”; G(x): “someone who loves x”)
5. Drop universal quantifiers::
   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$
6. CNF-ize propositionally:
   $(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

Conversion to CNF: Example

Consider: "Everyone who loves all animals is loved by someone"
$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Eliminate implications and biconditionals:

   $\forall x.(\neg[\forall y.(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$

2. Push inwards negations recursively:

   $\forall x.([\exists y.\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$
   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y.Loves(y,x)])$

3. Standardize variables:

   $\forall x.([\exists y.(Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z.Loves(z,x)])$

4. Skolemize:

   $\forall x.([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)])$
   (F(x): "an animal unloved by x"; G(x): "someone who loves x")

5. Drop universal quantifiers::

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

6. CNF-ize propositionally:

$(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

Remark: Bad Skolemization

Common mistake to avoid

- Do not:
  - apply Skolemization and/or
  - drop universal quantifiers

  before converting into NNF!
- Polarity of quantified subformulas affect Skolemization

$\Longrightarrow$ NNF-ization may convert $\exists$'s into $\forall$'s, and vice versa

Remark: Bad Skolemization

Common mistake to avoid

- Do not:
  - apply Skolemization and/or
  - drop universal quantifiers

  before converting into NNF!
- Polarity of quantified subformulas affect Skolemization

$\Longrightarrow$ NNF-ization may convert $\exists$'s into $\forall$'s, and vice versa

Bad Skolemization: Example

Wrong CNF-ization

$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [\exists y.Loves(y, x)])$

1. Too-early Skolemization & universal-quantifier dropping:
   $\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
   $([(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
2. NNF-ization and CNF-ization
   $([(Animal(y) \wedge \neg Loves(x, y))] \vee [Loves(G(x), x)])$
   $((Animal(y) \vee Loves(G(x), x)) \wedge ((\neg Loves(x, y)) \vee Loves(G(x), x)))$

"y" should be a Skolem function F(x) instead,
because "$\forall y.(...)$" occurred negatively
$\Longrightarrow$ should become "$\exists y.\neg(...)$", and hence y Skolemized into $F(x)$
(compare with previous slide)

Bad Skolemization: Example

Wrong CNF-ization

$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [\exists y.Loves(y, x)])$

1. Too-early Skolemization & universal-quantifier dropping:
   $\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
   $([(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
2. NNF-ization and CNF-ization
   $([(Animal(y) \wedge \neg Loves(x, y))] \vee [Loves(G(x), x)])$
   $((Animal(y) \vee Loves(G(x), x)) \wedge ((\neg Loves(x, y)) \vee Loves(G(x), x)))$

"y" should be a Skolem function F(x) instead,
because "$\forall y.(...)$" occurred negatively
$\Longrightarrow$ should become "$\exists y.\neg(...)$", and hence y Skolemized into $F(x)$
(compare with previous slide)

Bad Skolemization: Example

Wrong CNF-ization

$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [\exists y.Loves(y,x)])$

1. Too-early Skolemization & universal-quantifier dropping:
   $\forall x.([\forall y.(Animal(y) \rightarrow Loves(x,y))] \rightarrow [Loves(G(x),x)])$
   $([(Animal(y) \rightarrow Loves(x,y))] \rightarrow [Loves(G(x),x)])$
2. NNF-ization and CNF-ization
   $([(Animal(y) \wedge \neg Loves(x,y))] \vee [Loves(G(x),x)])$
   $((Animal(y) \vee Loves(G(x),x)) \wedge ((\neg Loves(x,y)) \vee Loves(G(x),x)))$

"y" should be a Skolem function F(x) instead,
because "$\forall y.(...)$" occurred negatively
$\Longrightarrow$ should become "$\exists y.\neg(...)$", and hence y Skolemized into $F(x)$
(compare with previous slide)

Bad Skolemization: Example

Wrong CNF-ization

$\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [\exists y.Loves(y, x)])$

1. Too-early Skolemization & universal-quantifier dropping:
   $\forall x.([\forall y.(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
   $([(Animal(y) \rightarrow Loves(x, y))] \rightarrow [Loves(G(x), x)])$
2. NNF-ization and CNF-ization
   $([(Animal(y) \wedge \neg Loves(x, y))] \vee [Loves(G(x), x)])$
   $((Animal(y) \vee Loves(G(x), x)) \wedge ((\neg Loves(x, y)) \vee Loves(G(x), x)))$

"y" should be a Skolem function F(x) instead,
because "$\forall y.(...)$" occurred negatively
$\Longrightarrow$ should become "$\exists y.\neg(...)$", and hence y Skolemized into $F(x)$
(compare with previous slide)

Outline

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\frac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generated $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generated $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\frac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\frac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\frac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\frac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Resolution

- FOL resolution rule, let $\theta \stackrel{\text{def}}{=} mgu(l_i, \neg m_j)$, s.t. $l_i\theta = \neg m_j\theta$:

$$\frac{(l_1 \vee ... \vee l_k) \qquad (m_1 \vee ... \vee m_n)}{(l_1 \vee ... \vee l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n)\theta}$$

- Ex: $\dfrac{Man(Socrates) \quad (\neg Man(x) \vee Mortal(x))}{Mortal(Socrates)}$ s.t. $\theta \stackrel{\text{def}}{=} \{x/Socrates\}$
- To prove that $\Gamma \models \alpha$ in FOL:
  - convert $\Gamma \wedge \neg\alpha$ to CNF
  - apply repeatedly resolution rule to $CNF(\Gamma \wedge \neg\alpha)$ until either
    - the empty clause is generate $\Longrightarrow \Gamma \models \alpha$
    - no more resolution step is applicable $\Longrightarrow \Gamma \not\models \alpha$
    - resource (time, memory) exhausted $\Longrightarrow$??
  - Hint: apply resolution first to unit clauses (unit resolution)
    - unit resolution alone complete for definite clauses
- Complete:
  - If there is a substitution θ such that $\Gamma \models \theta\alpha$, then it will return θ
  - If there is no such θ, then the procedure may not terminate
- Many strategies and tools available

Example: Resolution with Definite Clauses

KB: The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
Goal: Prove that Col. West is a criminal.

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as "hostile":
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow \neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow \neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as "hostile":
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow \neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as “hostile”:
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as "hostile":
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as "hostile":
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations:
  $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as “hostile”:
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...:
  $Enemy(Nono, America)$

Example: Resolution with Definite Clauses [cont.]

- it is a crime for an American to sell weapons to hostile nations: $\forall x, y, z.((American(x) \wedge Weapon(y) \wedge Hostile(z) \wedge Sells(x, y, z)) \rightarrow Criminal(x))$

$\Longrightarrow$ $\neg American(x) \vee \neg Weapon(y) \vee \neg Hostile(z) \vee \neg Sells(x, y, z) \vee Criminal(x)$

- Nono ... has some missiles
  $\exists x.(Owns(Nono, x) \wedge Missile(x)) \Longrightarrow Owns(Nono, M_1) \wedge Missile(M_1)$
- All of its missiles were sold to it by Colonel West
  $\forall x.((Missile(x) \wedge Owns(Nono, x)) \rightarrow Sells(West, x, Nono))$

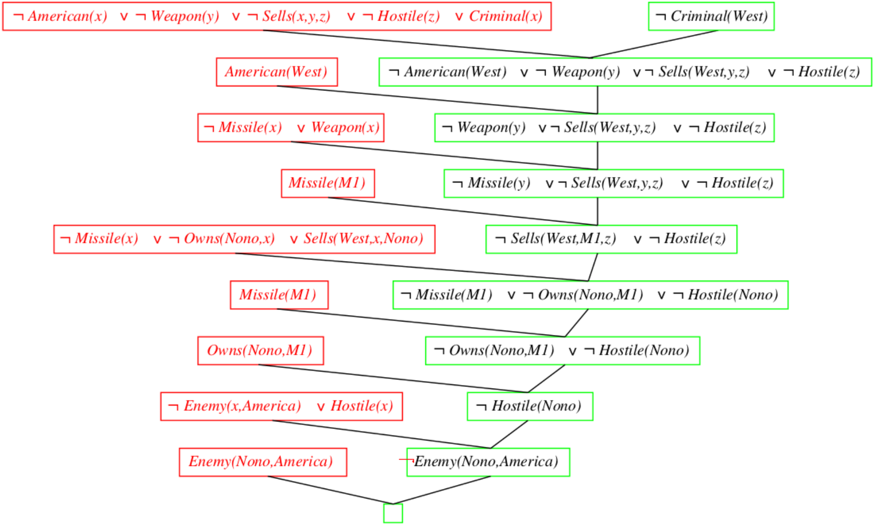
$\Longrightarrow$ $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

- Missiles are weapons:
  $\forall x.(Missile(x) \rightarrow Weapon(x)) \Longrightarrow \neg Missile(x) \vee Weapon(x)$
- An enemy of America counts as "hostile":
  $\forall x.(Enemy(x, America) \rightarrow Hostile(x))$

$\Longrightarrow$ $\neg Enemy(x, America) \vee Hostile(x)$

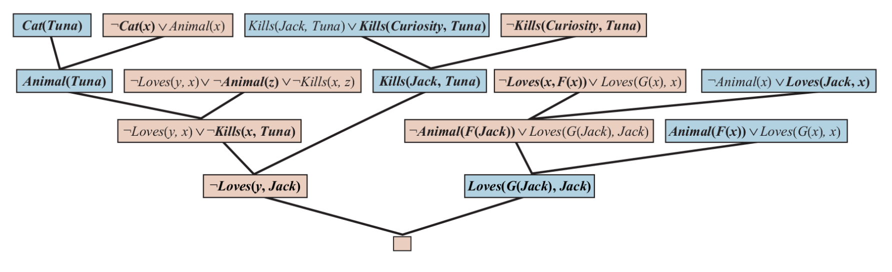
- West, who is American ...: $American(West)$
- The country Nono, an enemy of America ...: $Enemy(Nono, America)$

Example: Resolution with Definite Clauses



Example: Resolution with General Clauses

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?
(See AIMA book for FOL formalization and CNF-ization, or do it by exercise)



Resolution Strategies

Saturation Calculus:

- Given N_0 : set of (implicitly universally quantified) clauses.
- Derive $N_0, N_1, N_2, N_3, \ldots$ s.t. $N_{i+1} = N_i \cup \{C\}$,
  - where C is the conclusion of a resolution step from premises in N_i
- (under reasonable restrictions) is refutationally complete :

$$N_0 \models \bot \implies \bot \in N_i \text{ for some i}$$

Problem

- The resolution rule is prolific.
  - it generates many useless intermediate results
  - it may generate the same clauses in many different ways
- This motivates the introduction of resolution restrictions.

Resolution Strategies

Saturation Calculus:

- Given N_0 : set of (implicitly universally quantified) clauses.
- Derive N_0, N_1, N_2, N_3, ... s.t. $N_{i+1} = N_i \cup \{C\}$,
  - where C is the conclusion of a resolution step from premises in N_i
- (under reasonable restrictions) is refutationally complete :

$$N_0 \models \bot \implies \bot \in N_i \text{ for some i}$$

Problem

- The resolution rule is prolific.
  - it generates many useless intermediate results
  - it may generate the same clauses in many different ways
- This motivates the introduction of resolution restrictions.

Resolution Strategies

Saturation Calculus:

- Given N_0 : set of (implicitly universally quantified) clauses.
- Derive N_0, N_1, N_2, N_3, ... s.t. $N_{i+1} = N_i \cup \{C\}$,
  - where C is the conclusion of a resolution step from premises in N_i
- (under reasonable restrictions) is refutationally complete :

$$N_0 \models \bot \implies \bot \in N_i \text{ for some i}$$

Problem

- The resolution rule is prolific.
  - it generates many useless intermediate results
  - it may generate the same clauses in many different ways
- This motivates the introduction of resolution restrictions.

Resolution Strategies

Saturation Calculus:

- Given N_0 : set of (implicitly universally quantified) clauses.
- Derive $N_0, N_1, N_2, N_3, \ldots$ s.t. $N_{i+1} = N_i \cup \{C\}$,
  - where C is the conclusion of a resolution step from premises in N_i
- (under reasonable restrictions) is refutationally complete :

$$N_0 \models \bot \implies \bot \in N_i \text{ for some i}$$

Problem

- The resolution rule is prolific.
  - it generates many useless intermediate results
  - it may generate the same clauses in many different ways
- This motivates the introduction of resolution restrictions.

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \quad \cfrac{\cfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \qquad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \quad \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \quad \dfrac{\dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \quad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \quad \dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \frac{\dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \qquad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \frac{\dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \qquad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
-

$$Ex: \quad \cfrac{\cfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \quad P(A) \vee D}{R(A) \vee C \vee D}$$

- Multiple resolution steps are merged into one step
-

$$Ex: \quad \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \quad \dfrac{\dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \quad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \quad \dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \quad \frac{\dfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \quad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \quad \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Resolution Restrictions

Ordered resolution

- define stable atom ordering;
- resolve only maximal literals

Hyper-Resolution

- Clauses are divided into
  - "nuclei": those with ≥ 1 negative literals
  - "electrons" : those with positive literals only
- Resolution can occur only among one nucleus and one electron
- $$Ex: \quad \cfrac{\cfrac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C}{\neg P(A) \vee R(A) \vee C} \quad P(A) \vee D}{R(A) \vee C \vee D}$$
- Multiple resolution steps are merged into one step
- $$Ex: \quad \frac{\neg P(x) \vee \neg Q(x) \vee R(x) \quad Q(A) \vee C \quad P(A) \vee D}{R(A) \vee C \vee D}$$

$\Longrightarrow$ Globally, can produce only electrons

Exercise

- Solve the example of Colonel West using Hyper-Resolution strategy
- Solve the example of Curiosity & Tuna using Hyper-Resolution Strategy

Exercise

- Solve the example of Colonel West using Hyper-Resolution strategy
- Solve the example of Curiosity & Tuna using Hyper-Resolution Strategy

Outline

Dealing with Term Equalities [hints.]

To deal with equality formulas ($t_1 = t_2$)

- Combine resolution with Equal-term substitution rule
- Ex:

$$\dfrac{(4 \geq 3) \quad \dfrac{(S(x) = x+1) \quad (\neg(y \geq z) \vee (S(y) \geq S(z)))}{(\neg(y \geq z) \vee (y+1 \geq z+1))}}{4+1 \geq 3+1}$$

- Very inefficient
- Ad-hoc rules rule for equality: Paramodulation

Dealing with Term Equalities [hints.]

To deal with equality formulas ($t_1 = t_2$)

- Combine resolution with Equal-term substitution rule
- Ex:

$$\frac{(4 \geq 3) \quad \dfrac{(S(x)=x+1) \quad (\neg(y \geq z) \vee (S(y) \geq S(z)))}{(\neg(y \geq z) \vee (y+1 \geq z+1))}}{4+1 \geq 3+1}$$

- Very inefficient
- Ad-hoc rules rule for equality: Paramodulation

Dealing with Term Equalities [hints.]

To deal with equality formulas ($t_1 = t_2$)

- Combine resolution with Equal-term substitution rule
- Ex:

$$\dfrac{(4 \geq 3) \quad \dfrac{(S(x) = x+1) \quad (\neg(y \geq z) \vee (S(y) \geq S(z)))}{(\neg(y \geq z) \vee (y+1 \geq z+1))}}{4+1 \geq 3+1}$$

- Very inefficient
- Ad-hoc rules rule for equality: Paramodulation

Dealing with Term Equalities [hints.]

To deal with equality formulas $(t_1 = t_2)$

- Combine resolution with Equal-term substitution rule
- Ex:

$$(4 \geq 3) \frac{\dfrac{(S(x) = x+1) \quad (\neg(y \geq z) \vee (S(y) \geq S(z)))}{(\neg(y \geq z) \vee (y+1 \geq z+1))}}{4+1 \geq 3+1}$$

- Very inefficient
- Ad-hoc rules rule for equality: Paramodulation

Dealing with Term Equalities [hints.]

To deal with equality formulas ($t_1 = t_2$)

- Combine resolution with Equal-term substitution rule
- Ex:

$$(4 \geq 3)\frac{\dfrac{(S(x)=x+1) \quad (\neg(y \geq z) \vee (S(y) \geq S(z)))}{(\neg(y \geq z) \vee (y+1 \geq z+1))}}{4+1 \geq 3+1}$$

- Very inefficient
- Ad-hoc rules rule for equality: Paramodulation

Paramodulation

- Ground case:

$$\frac{D \vee (t = t') \quad C \vee L}{D \vee C \vee L\{t/t'\}} \quad \textit{literal}$$

- Example:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(a)}{R(b) \vee Q(c) \vee P(b)}$$

- General case:

$$\frac{D \vee (t = t') \quad C \vee L}{(D \vee C \vee L\{u/t'\})\theta} \quad \textit{where } \theta \stackrel{\text{def}}{=} mgu(t, u)$$

- Examples:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(x)}{R(b) \vee Q(c) \vee P(b)} \qquad \theta = \{x/a\}$$

$$\frac{R(g(c)) \vee (\overbrace{f(g(b))}^{t} = a) \quad Q(x) \vee P(g(\overbrace{f(x)}^{u}))}{R(g(c)) \vee Q(g(b)) \vee P(g(a))} \qquad \theta = \{x/g(b)\}$$

Paramodulation

- Ground case:

$$\frac{D \vee (t = t') \quad C \vee L}{D \vee C \vee L\{t/t'\}} \quad \textit{literal}$$

- Example:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(a)}{R(b) \vee Q(c) \vee P(b)}$$

- General case:

$$\frac{D \vee (t = t') \quad C \vee L}{(D \vee C \vee L\{u/t'\})\theta} \quad \textit{where } \theta \stackrel{\text{def}}{=} \textit{mgu}(t, u)$$

- Examples:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(x)}{R(b) \vee Q(c) \vee P(b)} \qquad \theta = \{x/a\}$$

$$\frac{R(g(c)) \vee (\overbrace{f(g(b))}^{t} = a) \quad Q(x) \vee P(g(\overbrace{f(x)}^{u}))}{R(g(c)) \vee Q(g(b)) \vee P(g(a))} \qquad \theta = \{x/g(b)\}$$

Paramodulation

- Ground case:
$$\frac{D \vee (t = t') \quad C \vee L}{D \vee C \vee L\{t/t'\}} \quad \textit{literal}$$
- Example:
$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(a)}{R(b) \vee Q(c) \vee P(b)}$$
- General case:
$$\frac{D \vee (t = t') \quad C \vee L}{(D \vee C \vee L\{u/t'\})\theta} \quad \textit{where } \theta \stackrel{\text{def}}{=} mgu(t, u)$$
- Examples:
$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(x)}{R(b) \vee Q(c) \vee P(b)} \qquad \theta = \{x/a\}$$
$$\frac{R(g(c)) \vee (\overbrace{f(g(b))}^{t} = a) \quad Q(x) \vee P(g(\overbrace{f(x)}^{u}))}{R(g(c)) \vee Q(g(b)) \vee P(g(a))} \qquad \theta = \{x/g(b)\}$$

Paramodulation

- Ground case:

$$\frac{D \vee (t = t') \quad C \vee L}{D \vee C \vee L\{t/t'\}} \quad \textit{literal}$$

- Example:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(a)}{R(b) \vee Q(c) \vee P(b)}$$

- General case:

$$\frac{D \vee (t = t') \quad C \vee L}{(D \vee C \vee L\{u/t'\})\theta} \quad \textit{where } \theta \stackrel{\text{def}}{=} mgu(t, u)$$

- Examples:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(x)}{R(b) \vee Q(c) \vee P(b)} \qquad \theta = \{x/a\}$$

$$\frac{R(g(c)) \vee (\overbrace{f(g(b))}^{t} = a) \quad Q(x) \vee P(g(\overbrace{f(x)}^{u}))}{R(g(c)) \vee Q(g(b)) \vee P(g(a))} \qquad \theta = \{x/g(b)\}$$

Paramodulation

- Ground case:

$$\frac{D \vee (t = t') \quad C \vee L}{D \vee C \vee L\{t/t'\}} \quad \textit{literal}$$

- Example:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(a)}{R(b) \vee Q(c) \vee P(b)}$$

- General case:

$$\frac{D \vee (t = t') \quad C \vee L}{(D \vee C \vee L\{u/t'\})\theta} \quad \textit{where } \theta \stackrel{\text{def}}{=} \textit{mgu}(t, u)$$

- Examples:

$$\frac{R(b) \vee (a = b) \quad Q(c) \vee P(x)}{R(b) \vee Q(c) \vee P(b)} \qquad \theta = \{x/a\}$$

$$\frac{R(g(c)) \vee (\overbrace{f(g(b))}^{t} = a) \quad Q(x) \vee P(g(\overbrace{f(x)}^{u}))}{R(g(c)) \vee Q(g(b)) \vee P(g(a))} \qquad \theta = \{x/g(b)\}$$

Outline

Example

Problem

Consider the following FOL formula set Γ:

1. $\forall x.\{[\forall y.(\mathrm{Child}(y) \rightarrow \mathrm{Loves}(x, y))] \rightarrow [\exists y.\mathrm{Loves}(y, x)]\}$
2. $\forall x.[\mathrm{Child}(x) \rightarrow \mathrm{Loves}(\mathrm{Mark}, x)]$
3. $\mathrm{Beats}(\mathrm{Mark}, \mathrm{Paul}) \vee \mathrm{Beats}(\mathrm{John}, \mathrm{Paul})$
4. $\mathrm{Child}(\mathrm{Paul})$
5. $\forall x.\{[\exists z.(\mathrm{Child}(z) \wedge \mathrm{Beats}(x, z))] \rightarrow [\forall y.\neg\mathrm{Loves}(y, x)]\}$

(*a*) Compute the CNF-ization of Γ, Skolemize & standardize variables

(*b*) Write a FOL-resolution inference of the query Beats(John, Paul) from the CNF-ized KB

Example

CNF-ization

(a) Compute the CNF-ization of Γ, Skolemize & standardize variables

1. $\forall x.\{[\forall y.(\text{Child}(y) \rightarrow \text{Loves}(x,y))] \rightarrow [\exists y.\text{Loves}(y,x)]\}$
   $\forall x.\{[\neg\forall y.(\text{Child}(y) \rightarrow \text{Loves}(x,y))] \vee [\exists y.\text{Loves}(y,x)]\}$
   $\forall x.\{[\exists y.(\text{Child}(y) \wedge \neg\text{Loves}(x,y))] \vee [\exists y.\text{Loves}(y,x)]\}$
   $\{[(\text{Child}(F(x)) \wedge \neg\text{Loves}(x,F(x)))] \vee [\text{Loves}(G(x),x)]\}$
   1. $\text{Child}(F(x)) \vee \text{Loves}(G(x),x)$
   2. $\neg\text{Loves}(y,F(y)) \vee \text{Loves}(G(y),y)$
2. $\neg\text{Child}(z) \vee \text{Loves}(\text{Mark},z)$
3. $\text{Beats}(\text{Mark},\text{Paul}) \vee \text{Beats}(\text{John},\text{Paul})$
4. $\text{Child}(\text{Paul})$
5. $\forall x.\{[\exists z.(\text{Child}(z) \wedge \text{Beats}(x,z))] \rightarrow [\forall y.\neg\text{Loves}(y,x)]\}$
   $\forall x.\{[\neg\exists z.(\text{Child}(z) \wedge \text{Beats}(x,z))] \vee [\forall y.\neg\text{Loves}(y,x)]\}$
   $\forall x.\{[\forall z.(\neg\text{Child}(z) \vee \neg\text{Beats}(x,z))] \vee [\forall y.\neg\text{Loves}(y,x)]\}$
   $\neg\text{Child}(z_2) \vee \neg\text{Beats}(x_2,z_2) \vee \neg\text{Loves}(y_2,x_2)$

where $F()$, $G()$ are Skolem unary functions.

Example

Resolution

(b) Write a FOL-resolution inference of the query Beats(John, Paul) from the CNF-ized KB:

6. $[1.2,\ 2.] \Longrightarrow \neg\text{Child}(F(\text{Mark})) \vee \text{Loves}(G(\text{Mark}), \text{Mark});$
7. $[1.1,\ 6.] \Longrightarrow \text{Loves}(G(\text{Mark}), \text{Mark});$
8. $[4,\ 5.] \Longrightarrow \neg\text{Beats}(x_2, \text{Paul}) \vee \neg\text{Loves}(y_2, x_2);$
9. $[7,\ 8.] \Longrightarrow \neg\text{Beats}(\text{Mark}, \text{Paul});$
10. $[3,\ 9.] \Longrightarrow \text{Beats}(\text{John}, \text{Paul});$