CS228 : Logic for Computer Science

Module 3: Predicate Logic

Lecture 12: Syntax of Predicate Logic

Instructor: S. Akshay

IIT Bombay, India

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

1. Propositional Resolution - soundness, termination and completeness

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

1. Propositional Resolution - soundness, termination and completeness
2. The DPLL algorithm

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

1. Propositional Resolution - soundness, termination and completeness
2. The DPLL algorithm
3. Clause learning, the CDCL algorithm, and modern SAT solvers

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction, Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

1. Propositional Resolution - soundness, termination and completeness
2. The DPLL algorithm
3. Clause learning, the CDCL algorithm, and modern SAT solvers
4. A tool demo of SAT/SMT solvers in practice

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?
  - Being a student/teacher
  - Being younger
  - Every/some – these are quantifiers

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?
  - Being a student/teacher : these are called predicates.
    - e.g., S(Osim) will denote Osim is a student, T(Rowena) will denote Rowena is a teacher.
  - Being younger : these are called predicates.
    - e.g., Y(Osim, Rowena) will denote Osim is younger than Rowena
  - Every/some – these are quantifiers

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?
  - Being a student/teacher : these are called unary predicates.
    - e.g., S(Osim) will denote Osim is a student, T(Rowena) will denote Rowena is a teacher.
  - Being younger : these are called binary predicates.
    - e.g., Y(Osim, Rowena) will denote Osim is younger than Rowena
  - Every/some – these are quantifiers

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?
  - Being a student/teacher : these are called unary predicates.
    - e.g., S(Osim) will denote Osim is a student, T(Rowena) will denote Rowena is a teacher.
  - Being younger : these are called binary predicates.
    - e.g., Y(Osim, Rowena) will denote Osim is younger than Rowena
  - Every/some – these are quantifiers
  - Variables – to be able to say "$\forall\ x\ S(x)$" instead of writing each student out...

Next Module: Predicate Logic

Propositional Logic was about simple statements and connectives

- e.g., If Osim is a student and Rowena is a teacher, then Osim is younger than Rowena.

Predicate logic goes beyond and talks about quantifiers and predicates

- e.g., Every student is younger than some teacher.
- What are the constituents of this sentence?
  - Being a student/teacher : these are called unary predicates.
    - e.g., S(Osim) will denote Osim is a student, T(Rowena) will denote Rowena is a teacher.
  - Being younger : these are called binary predicates.
    - e.g., Y(Osim, Rowena) will denote Osim is younger than Rowena
  - Every/some – these are quantifiers
  - Variables – to be able to say "$\forall\ x\ S(x)$" instead of writing each student out...
- e.g., $\forall x(S(x) \rightarrow \exists y(T(y) \wedge Y(x,y)))$ For all x, if x is a student, then there is some y which is a teacher such that x is younger than y.

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

What is the difference between these two statements? Are they "equivalent"?

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

What is the difference between these two statements? Are they "equivalent"?

Can we show they are equivalent?

1. Need to define semantics, and semantical equivalence.

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

What is the difference between these two statements? Are they "equivalent"?

Can we show they are equivalent?

1. Need to define semantics, and semantical equivalence.
2. Can we also show this using proof rules? Need Natural deduction system!

Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \rightarrow A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

What is the difference between these two statements? Are they "equivalent"?

Can we show they are equivalent?

1. Need to define semantics, and semantical equivalence.
2. Can we also show this using proof rules? Need Natural deduction system!
3. First, what is the syntax!?

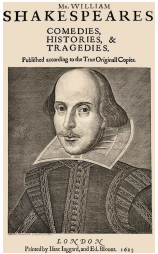
Constituents of Predicate logic

Exercise 12.1 Convert the following sentence into Predicate logic.

Not all that glitters is Gold.

- Choose predicates: $G(x)$: x glitters, $A(y)$: y is gold.
- $\neg(\forall x(G(x) \to A(x)))$
- $\exists x(G(x) \wedge \neg A(x))$

What is the difference between these two statements? Are they "equivalent"?



Can we show they are equivalent?

1. Need to define semantics, and semantical equivalence.
2. Can we also show this using proof rules? Need Natural deduction system!
3. First, what is the syntax!?

Fun fact: this proverb is a common misquote of Shakespeare's The Merchant of Venice – the original line is "All that glisters is not gold."

Topic 12.1

Syntax of Predicate Logic

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding!

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$
- Function symbols are very useful to get compact encodings.

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$
- Function symbols are very useful to get compact encodings.
- Only applicable when no ambiguity, e.g., father of. Can't use for "brother of". (why?)

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$
- Function symbols are very useful to get compact encodings.
- Only applicable when no ambiguity, e.g., father of. Can't use for "brother of". (why?)
- Can take multiple or zero arguments. Functions with zero arguments are called constants.

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$
- Function symbols are very useful to get compact encodings.
- Only applicable when no ambiguity, e.g., father of. Can't use for "brother of". (why?)
- Can take multiple or zero arguments. Functions with zero arguments are called constants.
- Thus, a and ℓ above are actually constants, not variables!

Another difference from Propositional Logic

Exercise 12.2 Encode into Predicate logic.

Asif and Lavanya have the same grandfather.

- Use "variables" a for "Asif", ℓ for "Lavanya"
- Binary predicate F for "being father", i.e., $F(x, y)$ if y is father of x.

Function Symbols: Suppose we had a function symbol f(x) to mean father of x.

- Then we get an easy encoding! $f(f(a)) = f(f(\ell))$
- Function symbols are very useful to get compact encodings.
- Only applicable when no ambiguity, e.g., father of. Can't use for "brother of". (why?)
- Can take multiple or zero arguments. Functions with zero arguments are called constants.
- Thus, a and ℓ above are actually constants, not variables!

Important note: Equality $=$ is a special binary predicate symbol. Always assumed to be present!

Our Alphabet

Variable names: x, y, z

Our Alphabet

Variable names: x, y, z

- Function symbols: $f, g, h...$
  - Arity of function is number of arguments.
  - 0-ary functions are constants, we will use $a, b, c...$
- Relation (Predicate) symbols: $P, Q, R, ...$
  - Hence, called Predicate logic or calculus.
  - Arity of predicate is number of arguments.

Our Alphabet

Variable names: x, y, z

- Function symbols: $f, g, h...$
  - Arity of function is number of arguments.
  - 0-ary functions are constants, we will use $a, b, c...$
- Relation (Predicate) symbols: $P, Q, R, ...$ (including $=$)
  - Hence, called Predicate logic or calculus.
  - Arity of predicate is number of arguments.

Our Alphabet

Variable names: x, y, z

Vocabulary

- Function symbols: $f, g, h...$
  - Arity of function is number of arguments.
  - 0-ary functions are constants, we will use $a, b, c...$
- Relation (Predicate) symbols: $P, Q, R, ...$ (including $=$)
  - Hence, called Predicate logic or calculus.
  - Arity of predicate is number of arguments.

Our Alphabet

Variable names: x, y, z

Vocabulary

- Function symbols: $f, g, h...$
  - Arity of function is number of arguments.
  - 0-ary functions are constants, we will use $a, b, c...$
- Relation (Predicate) symbols: $P, Q, R, ...$ (including $=$)
  - Hence, called Predicate logic or calculus.
  - Arity of predicate is number of arguments.

Fixed symbols

- All symbols from prop. logic: $\wedge, \vee, \neg, \rightarrow, (,)$
- New ones: $\exists, \forall$ (quantifiers)

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined,

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Terms

- every variable and constant is a term
- if f is an n-ary function and $t_1, \ldots t_n$ are terms, then so is $f(t_1, \ldots, t_n)$.

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Terms

- every variable and constant is a term
- if f is an n-ary function and $t_1, \ldots t_n$ are terms, then so is $f(t_1, \ldots, t_n)$.

$$t ::= x \mid c \mid f(t, \ldots, t)$$

where x is over variables, c over constants and f over other function symbols.

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Terms

- every variable and constant is a term
- if f is an n-ary function and $t_1, \ldots t_n$ are terms, then so is $f(t_1, \ldots, t_n)$.

$$t ::= x \mid c \mid f(t, \ldots, t)$$

where x is over variables, c over constants and f over other function symbols.

Examples (recall Exercise 12.2)

- a (Asif), ℓ (Lavanya) are terms – they are constants.

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Terms

- every variable and constant is a term
- if f is an n-ary function and $t_1, \ldots t_n$ are terms, then so is $f(t_1, \ldots, t_n)$.

$$t ::= x \mid c \mid f(t, \ldots, t)$$

where x is over variables, c over constants and f over other function symbols.

Examples (recall Exercise 12.2)

- a (Asif), ℓ (Lavanya) are terms – they are constants.
- $f(a)$ (father of Asif), $f(f(a))$ (grandfather of Asif) are terms too!

Syntax of Predicate Logic: Terms

- Strings over the alphabet defined, but not any strings, only those with special structure!
- Two classes of syntactic objects: terms and formulas.

Terms

- every variable and constant is a term
- if f is an n-ary function and $t_1, \ldots t_n$ are terms, then so is $f(t_1, \ldots, t_n)$.

$$t ::= x \mid c \mid f(t, \ldots, t)$$

where x is over variables, c over constants and f over other function symbols.

Examples (recall Exercise 12.2)

- a (Asif), ℓ (Lavanya) are terms – they are constants.
- $f(a)$ (father of Asif), $f(f(a))$ (grandfather of Asif) are terms too!
- Careful: S(Osim) (recall the earlier example!) is *not* a term!
  - f is a function symbol: maps objects to objects, so $f(a)$ *denotes an object* – a term.
  - S is a predicate symbol: maps objects to truth values, so S(Osim) is True or False – not a term!

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Formulas

- If R is an m-ary predicate, $t_1, \ldots t_m$ are terms then $R(t_1, \ldots t_m)$ is a(n atomic) formula
- Boolean connectives give formulas: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ etc.
- Quantifiers give formulas: if φ is a formula, x a variable, then so is $\exists x\varphi$, $\forall x\varphi$.

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Formulas

- If R is an m-ary predicate, $t_1, \ldots t_m$ are terms then $R(t_1, \ldots t_m)$ is a(n atomic) formula
- Boolean connectives give formulas: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ etc.
- Quantifiers give formulas: if φ is a formula, x a variable, then so is $\exists x\varphi$, $\forall x\varphi$.

$$\varphi ::= P(t_1, \ldots t_n) \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x\varphi) \mid (\exists x\varphi)$$

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Formulas

- If R is an m-ary predicate, $t_1, \ldots t_m$ are terms then $R(t_1, \ldots t_m)$ is a(n atomic) formula
- Boolean connectives give formulas: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ etc.
- Quantifiers give formulas: if φ is a formula, x a variable, then so is $\exists x\varphi$, $\forall x\varphi$.

$$\varphi ::= P(t_1, \ldots t_n) \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\forall x\varphi) \mid (\exists x\varphi)$$

Examples

- $f(f(a)) = f(f(\ell))$ is an (atomic) formula – Asif and Lavanya have the same grandfather!

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Formulas

- If R is an m-ary predicate, $t_1, \ldots t_m$ are terms then $R(t_1, \ldots t_m)$ is a(n atomic) formula
- Boolean connectives give formulas: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ etc.
- Quantifiers give formulas: if φ is a formula, x a variable, then so is $\exists x\varphi$, $\forall x\varphi$.

$$\varphi ::= P(t_1, \ldots t_n) \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\forall x\varphi) \mid (\exists x\varphi)$$

Examples

- $f(f(a)) = f(f(\ell))$ is an (atomic) formula – Asif and Lavanya have the same grandfather!
- $\forall x(S(x) \to \exists y(T(y) \wedge Y(x, y)))$ is a formula – every student is younger than some teacher.

Syntax of Predicate Logic: Formulas

An expression that (informally) *evaluates to a truth value* is called a formula!

Formulas

- If R is an m-ary predicate, $t_1, \ldots t_m$ are terms then $R(t_1, \ldots t_m)$ is a(n atomic) formula
- Boolean connectives give formulas: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ etc.
- Quantifiers give formulas: if φ is a formula, x a variable, then so is $\exists x\varphi$, $\forall x\varphi$.

$$\varphi ::= P(t_1, \ldots t_n) \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\forall x\varphi) \mid (\exists x\varphi)$$

Examples

- $f(f(a)) = f(f(\ell))$ is an (atomic) formula – Asif and Lavanya have the same grandfather!
- $\forall x(S(x) \to \exists y(T(y) \wedge Y(x, y)))$ is a formula – every student is younger than some teacher.

Why "First Order"?

Called First order logic because quantification is only over individual variables...

Binding and Parse trees

Binding rules as expected!

$$\{\neg, \exists x, \forall x\} \quad > \quad \{\vee, \wedge\} \quad > \quad \{\rightarrow\}$$

Binding and Parse trees

Binding rules as expected!

$$\{\neg, \exists x, \forall x\} \quad > \quad \{\vee, \wedge\} \quad > \quad \{\rightarrow\}$$

- $\rightarrow$ is also right-associative: $\varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3$ means $\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$.
- Unlike $\wedge, \vee$ (where bracketing never changes the meaning), bracketing genuinely changes the truth value for $\rightarrow$!

Binding and Parse trees

Binding rules as expected!

$$\{\neg, \exists x, \forall x\} \quad > \quad \{\vee, \wedge\} \quad > \quad \{\rightarrow\}$$

- $\rightarrow$ is also right-associative: $\varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3$ means $\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$.
- Unlike $\wedge, \vee$ (where bracketing never changes the meaning), bracketing genuinely changes the truth value for $\rightarrow$!

Formulas can be represented as Parse trees as well!

Binding and Parse trees

Binding rules as expected!

$$\{\neg, \exists x, \forall x\} \quad > \quad \{\vee, \wedge\} \quad > \quad \{\rightarrow\}$$

- $\rightarrow$ is also right-associative: $\varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3$ means $\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$.
- Unlike $\wedge, \vee$ (where bracketing never changes the meaning), bracketing genuinely changes the truth value for $\rightarrow$!

Formulas can be represented as Parse trees as well! We omit brackets when they do not introduce ambiguity.

Binding and Parse trees

Binding rules as expected!

$$\{\neg, \exists x, \forall x\} \quad > \quad \{\vee, \wedge\} \quad > \quad \{\rightarrow\}$$

- $\rightarrow$ is also right-associative: $\varphi_1 \rightarrow \varphi_2 \rightarrow \varphi_3$ means $\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$.
- Unlike $\wedge, \vee$ (where bracketing never changes the meaning), bracketing genuinely changes the truth value for $\rightarrow$!

Formulas can be represented as Parse trees as well! We omit brackets when they do not introduce ambiguity.

Exercise 12.3

1. Draw the Parse tree for $\forall x((P(x) \rightarrow Q(x)) \wedge S(x, y))$.
2. Encode into Predicate logic and draw its Parse tree: every son of my father is my brother.

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

$\forall x$

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

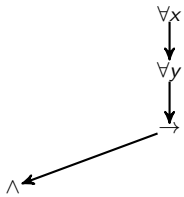
$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

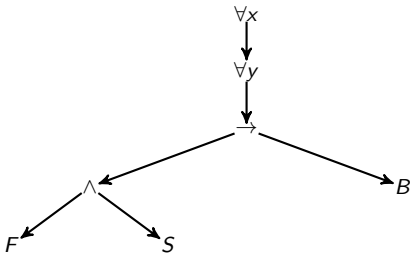


Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

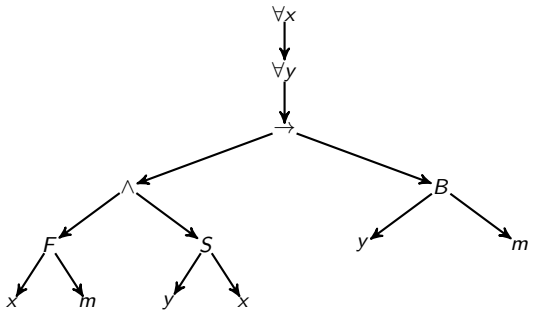


Encoding "every son of my father is my brother"

Using Predicates Only:

$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

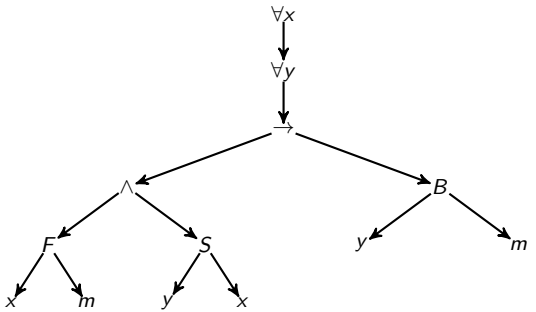
$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



Encoding "every son of my father is my brother"

Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$

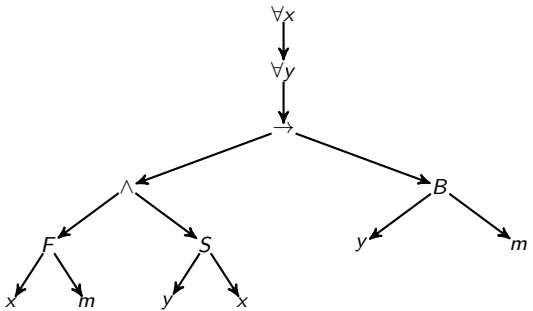


Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

Encoding "every son of my father is my brother"

Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



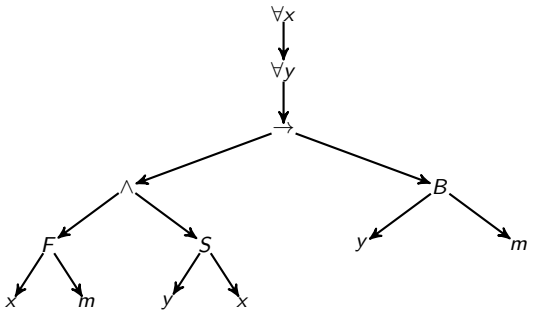
Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$

Encoding "every son of my father is my brother"

Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



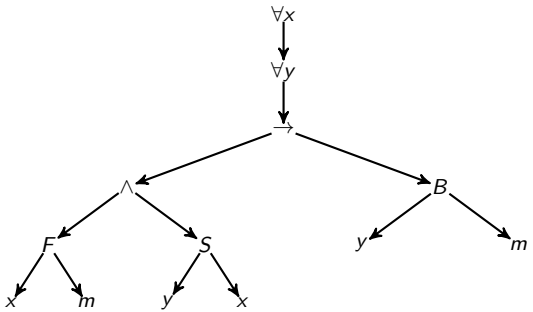
Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$
$$\forall x$$

Encoding "every son of my father is my brother"

Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



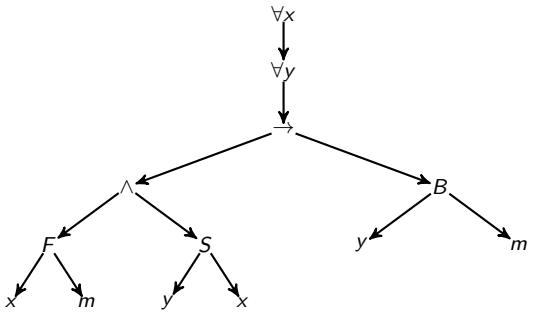
Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$

Encoding "every son of my father is my brother"

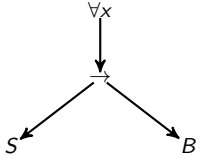
Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

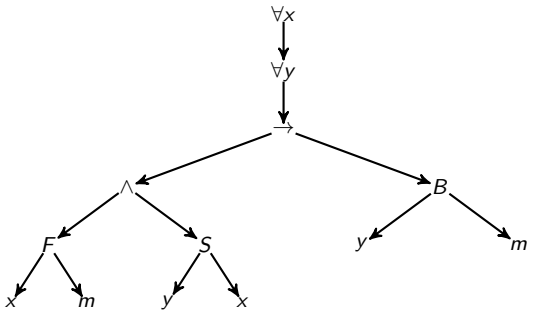
$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$



Encoding "every son of my father is my brother"

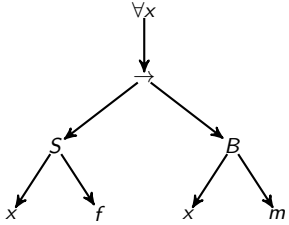
Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

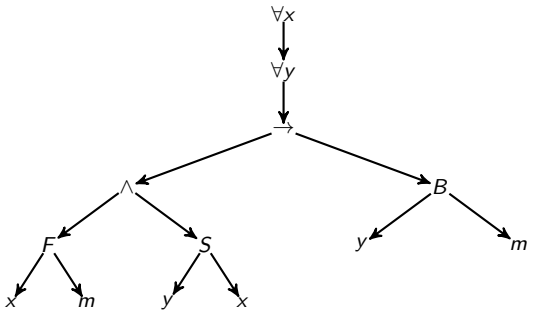
$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$



Encoding "every son of my father is my brother"

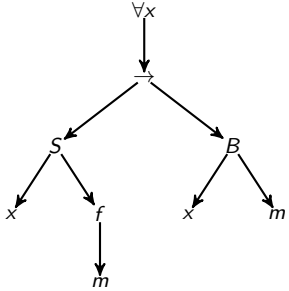
Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

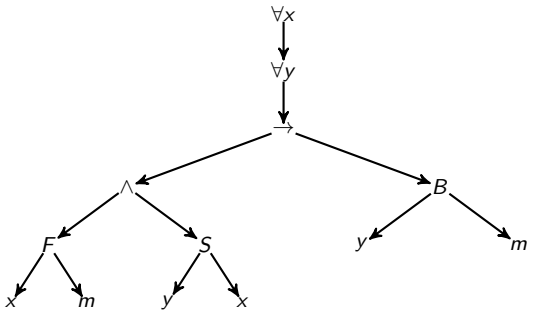
$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$



Encoding "every son of my father is my brother"

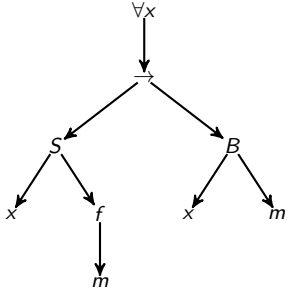
Using Predicates Only:
$F(x, y)$: x father of y; $S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x \forall y (F(x, m) \wedge S(y, x) \rightarrow B(y, m))$$



Using a Function Symbol (reuse f from Ex 12.2!):
$S(x, y)$: x son of y; $B(x, y)$: x brother of y; m: "me".

$$\forall x (S(x, f(m)) \rightarrow B(x, m))$$



Much more compact than the predicates-only version – this is exactly why function symbols are useful!

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $\mathit{free}(\varphi)$.

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $\mathit{free}(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $\mathit{free}(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $\mathit{free}(\varphi) = \mathit{free}(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi_1) \cup \mathit{free}(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi) \setminus \{x\}$

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $\mathit{free}(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $\mathit{free}(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $\mathit{free}(\varphi) = \mathit{free}(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi_1) \cup \mathit{free}(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi) \setminus \{x\}$

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $free(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $free(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $free(\varphi) = free(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $free(\varphi) = free(\psi_1) \cup free(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $free(\varphi) = free(\psi) \setminus \{x\}$

If φ has free variables $\{x, y\}$, then we write $\varphi(x, y)$.
A formula with no free variables is called a sentence, e.g., $\exists x \forall y\ f(x) = y$.

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $free(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $free(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $free(\varphi) = free(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $free(\varphi) = free(\psi_1) \cup free(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $free(\varphi) = free(\psi) \setminus \{x\}$

If φ has free variables $\{x, y\}$, then we write $\varphi(x, y)$.
A formula with no free variables is called a sentence, e.g., $\exists x \forall y\ f(x) = y$.

Exercise 12.4

1. What is free in $\psi := ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$? Is ψ a sentence?

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $\mathit{free}(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $\mathit{free}(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $\mathit{free}(\varphi) = \mathit{free}(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi_1) \cup \mathit{free}(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi) \setminus \{x\}$

If φ has free variables $\{x, y\}$, then we write $\varphi(x, y)$.
A formula with no free variables is called a sentence, e.g., $\exists x \forall y\ f(x) = y$.

Exercise 12.4

1. What is free in $\psi := ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$? Is ψ a sentence?
2. A variable is bound if it is quantified in a formula. Give a recursive definition of $\mathit{bnd}(\varphi)$.

Free variables, Bound variables and Sentences

Free variables are those that are not quantified in a formula, denoted $\mathit{free}(\varphi)$.

A recursive definition

- if φ is an atomic formula, then $\mathit{free}(\varphi) = \{x \mid x \text{ occurs in } \varphi\}$.
- if $\varphi = \neg\psi$, then $\mathit{free}(\varphi) = \mathit{free}(\psi)$.
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi_1) \cup \mathit{free}(\psi_2)$.
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $\mathit{free}(\varphi) = \mathit{free}(\psi) \setminus \{x\}$

If φ has free variables $\{x, y\}$, then we write $\varphi(x, y)$.
A formula with no free variables is called a sentence, e.g., $\exists x \forall y\ f(x) = y$.

Exercise 12.4

1. What is free in $\psi := ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$? Is ψ a sentence?
2. A variable is bound if it is quantified in a formula. Give a recursive definition of $\mathit{bnd}(\varphi)$.
3. Are free and bound variables complements? Why or why not?

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!
Consider again $\psi = ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$.

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!

Consider again $\psi = ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$.

$bnd(\psi) = bnd(\exists x\ P(x, y)) \cup bnd(\forall y\ Q(x, y))$

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!

Consider again $\psi = ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$.

$$bnd(\psi) = bnd(\exists x\ P(x, y)) \cup bnd(\forall y\ Q(x, y))$$
$$= (bnd(P(x, y)) \cup \{x\}) \cup (bnd(Q(x, y)) \cup \{y\})$$

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!

Consider again $\psi = ((\exists x\ P(x, y)) \wedge (\forall y\ Q(x, y)))$.

$$bnd(\psi) = bnd(\exists x\ P(x, y)) \cup bnd(\forall y\ Q(x, y))$$
$$= (bnd(P(x, y)) \cup \{x\}) \cup (bnd(Q(x, y)) \cup \{y\})$$
$$= (\emptyset \cup \{x\}) \cup (\emptyset \cup \{y\}) = \{x, y\}$$

Free and Bound Variables: Completing the Definition

Recursive definition of $bnd(\varphi)$, alongside $free(\varphi)$

- if φ is an atomic formula, then $bnd(\varphi) = \emptyset$
- if $\varphi = \neg\psi$, then $bnd(\varphi) = bnd(\psi)$
- if $\varphi = \psi_1 \circ \psi_2$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, then $bnd(\varphi) = bnd(\psi_1) \cup bnd(\psi_2)$
- if $\varphi = Qx\ \psi$ for $Q \in \{\exists, \forall\}$, then $bnd(\varphi) = bnd(\psi) \cup \{x\}$

Free and Bound variables are not complements of each other!

Consider again $\psi = ((\exists x\ P(x,y)) \wedge (\forall y\ Q(x,y)))$.

$$bnd(\psi) = bnd(\exists x\ P(x,y)) \cup bnd(\forall y\ Q(x,y))$$
$$= (bnd(P(x,y)) \cup \{x\}) \cup (bnd(Q(x,y)) \cup \{y\})$$
$$= (\emptyset \cup \{x\}) \cup (\emptyset \cup \{y\}) = \{x, y\}$$

$free(\psi) = bnd(\psi) = \{x, y\}$

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$
- Problem: the y from t is now captured by $\exists y$ – it's no longer free like it was in t!

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$
- Problem: the y from t is now captured by $\exists y$ – it's no longer free like it was in t!

Definition

Term t is free for x in φ if no free occurrence of x in φ is in the "scope" of $\forall y$ or $\exists y$ for any var y occurring in t.

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$
- Problem: the y from t is now captured by $\exists y$ – it's no longer free like it was in t!

Definition

Term t is free for x in φ if no free occurrence of x in φ is in the "scope" of $\forall y$ or $\exists y$ for any var y occurring in t. e.g., $f(z, x)$ is free for x in ψ, but $f(y, x)$ is not.

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$
- Problem: the y from t is now captured by $\exists y$ – it's no longer free like it was in t!

Definition

Term t is free for x in φ if no free occurrence of x in φ is in the "scope" of $\forall y$ or $\exists y$ for any var y occurring in t. e.g., $f(z, x)$ is free for x in ψ, but $f(y, x)$ is not.

Substitution

$\varphi[t/x]$ is formula obtained by replacing each free occurrence of x in φ by t, if t is free for x in φ.

Substitution in FOL

Variables are placeholders; so we must be able to replace them with concrete terms: substitution.

- Suppose $x \in \mathit{free}(\varphi)$ and t is any term.
- Can we replace every free occurrence of x in φ with t?
- For example, consider $\psi = \exists y R(x, y) \vee \forall x R(z, x)$.
- If we naively try substituting $t = f(y, x)$ for x in ψ, we get $\exists y R(f(y, x), y) \vee \forall x R(z, x)$
- Problem: the y from t is now captured by $\exists y$ – it's no longer free like it was in t!

Definition

Term t is free for x in φ if no free occurrence of x in φ is in the "scope" of $\forall y$ or $\exists y$ for any var y occurring in t. e.g., $f(z, x)$ is free for x in ψ, but $f(y, x)$ is not.

Substitution

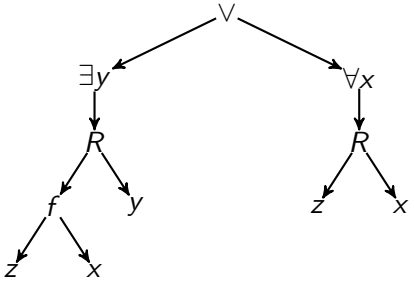
$\varphi[t/x]$ is formula obtained by replacing each free occurrence of x in φ by t, if t is free for x in φ.

For ψ defined above: $\psi[f(z, x)/x] := \exists y R(f(z, x), y) \vee \forall x R(z, x)$

Substitution in FOL: Correct vs. Captured

Correct: $\psi[f(z,x)/x]$

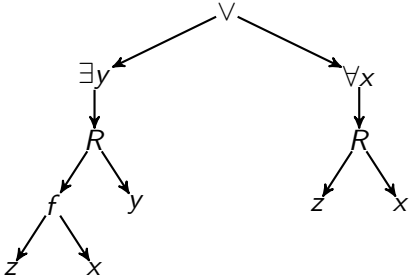
$\exists y\, R(f(z,x),y) \vee \forall x\, R(z,x)$



Substitution in FOL: Correct vs. Captured

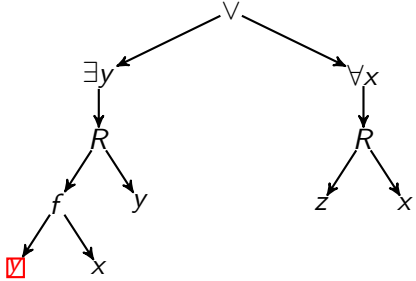
Correct: $\psi[f(z,x)/x]$

$\exists y\, R(f(z,x),y) \vee \forall x\, R(z,x)$



Wrong: $\psi[f(y,x)/x]$

$\exists y\, R(f(y,x),y) \vee \forall x\, R(z,x)$

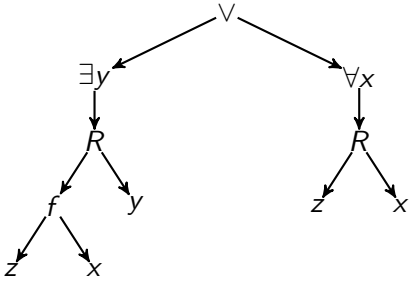


The y inside $f(y,x)$ (boxed in red) is now captured by $\exists y$ – it's a different leaf than the term's original free y, but shares its scope!

Substitution in FOL: Correct vs. Captured

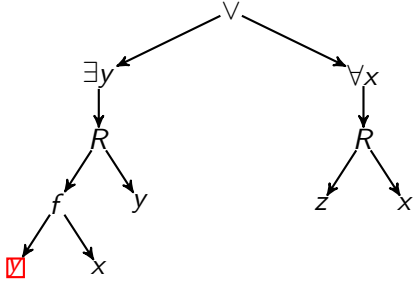
Correct: $\psi[f(z,x)/x]$

$\exists y\, R(f(z,x),y) \vee \forall x\, R(z,x)$



Wrong: $\psi[f(y,x)/x]$

$\exists y\, R(f(y,x),y) \vee \forall x\, R(z,x)$



The y inside $f(y,x)$ (boxed in red) is now captured by $\exists y$ – it's a different leaf than the term's original free y, but shares its scope!

- Fix: rename the bound y to a fresh y' first
- Rewrite ψ as $\exists y'\, R(x,y') \vee \forall x\, R(z,x)$ (same meaning, y is just a placeholder), *then* substitute.