Propositional Language - Syntax

Lila Kari
Based in part on materials and comments by Borzoo Bonakdarpour, Daniela Maftuleac, Peter Van Beek, Jonathan Buss, Collin Roberts, Richard Trefler, Stephen Watt

School of Computer Science
University of Waterloo
Waterloo, Canada

1. **Syntax of propositional language $\mathcal{L}^p$: atoms, formulas**
2. Generating and parsing formulas of $\mathcal{L}^p$
3. Mathematical induction (optional), structural induction
4. Unique Readability Theorem
5. Precedence rules for connectives, scope of a connective

Propositional language

- By using connectives, we can combine propositions, whether they are atomic themselves or compound.

Propositional language

- By using connectives, we can combine propositions, whether they are atomic themselves or compound.
- To prevent ambiguity, we will introduce fully parenthesized expressions that can be parsed in a unique way. (Later, we will use precedence rules to omit some parentheses.)

Propositional language

- By using connectives, we can combine propositions, whether they are atomic themselves or compound.
- To prevent ambiguity, we will introduce fully parenthesized expressions that can be parsed in a unique way. (Later, we will use precedence rules to omit some parentheses.)
- In the following, we construct the propositional language $\mathcal{L}^p$, which is the formal language of propositional logic.

Propositional language

- By using connectives, we can combine propositions, whether they are atomic themselves or compound.
- To prevent ambiguity, we will introduce fully parenthesized expressions that can be parsed in a unique way. (Later, we will use precedence rules to omit some parentheses.)
- In the following, we construct the propositional language $\mathcal{L}^p$, which is the formal language of propositional logic.
- The set of formulas in $\mathcal{L}^p$, denoted by $\text{Form}(\mathcal{L}^p)$, will then be defined by a set of formation rules which produce expressions (string of symbols) in $\mathcal{L}^p$ belonging to $\text{Form}(\mathcal{L}^p)$.

Propositional language

- By using connectives, we can combine propositions, whether they are atomic themselves or compound.
- To prevent ambiguity, we will introduce fully parenthesized expressions that can be parsed in a unique way. (Later, we will use precedence rules to omit some parentheses.)
- In the following, we construct the propositional language $\mathcal{L}^p$, which is the formal language of propositional logic.
- The set of formulas in $\mathcal{L}^p$, denoted by $\text{Form}(\mathcal{L}^p)$, will then be defined by a set of formation rules which produce expressions (string of symbols) in $\mathcal{L}^p$ belonging to $\text{Form}(\mathcal{L}^p)$.
- If an expression in $\mathcal{L}^p$ can be produced by the formation rules then it is a formula in $\text{Form}(\mathcal{L}^p)$, otherwise it is not a formula in $\text{Form}(\mathcal{L}^p)$.

Syntax of the propositional language $\mathcal{L}^p$

$\mathcal{L}^p$ is the formal language of propositional logic. Strings in $\mathcal{L}^p$ comprise three classes of symbols:

Syntax of the propositional language $\mathcal{L}^p$

$\mathcal{L}^p$ is the formal language of propositional logic. Strings in $\mathcal{L}^p$ comprise three classes of symbols:

- proposition symbols: $p, q, r, ...$, with or without subscripts
  (note that there is an unbounded number of such symbols)

Syntax of the propositional language $\mathcal{L}^p$

$\mathcal{L}^p$ is the formal language of propositional logic. Strings in $\mathcal{L}^p$ comprise three classes of symbols:

- proposition symbols: $p, q, r, ...$, with or without subscripts (note that there is an unbounded number of such symbols)
- connective symbols: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

	Oral reading of connectives	
$\neg$	not	negation
$\wedge$	and	conjunction
$\vee$	inclusive or	disjunction
$\rightarrow$	if, then (implies)	implication
$\leftrightarrow$	equivalent to (iff)	equivalence

Syntax of the propositional language $\mathcal{L}^p$

$\mathcal{L}^p$ is the formal language of propositional logic. Strings in $\mathcal{L}^p$ comprise three classes of symbols:

- proposition symbols: $p, q, r, ...$, with or without subscripts (note that there is an unbounded number of such symbols)
- connective symbols: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

	Oral reading of connectives	
$\neg$	not	negation
$\wedge$	and	conjunction
$\vee$	inclusive or	disjunction
$\rightarrow$	if, then (implies)	implication
$\leftrightarrow$	equivalent to (iff)	equivalence

- punctuation symbols: left and right parentheses : $($, $)$.

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it
  – for the expressions above, lengths are $1, 2, 3, 4$ and 6.

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it
  – for the expressions above, lengths are $1, 2, 3, 4$ and 6.
- Empty expression: expression of length zero, denoted by ϵ (not to be confused with the empty set $\emptyset$).

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it
  – for the expressions above, lengths are $1, 2, 3, 4$ and 6.
- Empty expression: expression of length zero, denoted by ϵ (not to be confused with the empty set $\emptyset$).
  - The textbook denotes the empty expression by $\emptyset$, but we will write ϵ instead, to prevent confusion.

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it
  – for the expressions above, lengths are $1, 2, 3, 4$ and 6.
- Empty expression: expression of length zero, denoted by ϵ (not to be confused with the empty set $\emptyset$).
  - The textbook denotes the empty expression by $\emptyset$, but we will write ϵ instead, to prevent confusion.
- Two expressions U and V are equal iff they are of the same length and have the same symbols in the same order.

Expressions of $\mathcal{L}^p$

- Expressions are finite strings of the allowed symbols
  – $p, pq, (r), p\wedge \rightarrow q$ and $\neg(p \vee q)$ are expressions in $\mathcal{L}^p$.
- Length of expression: the number of occurrences of symbols in it
  – for the expressions above, lengths are $1, 2, 3, 4$ and 6.
- Empty expression: expression of length zero, denoted by ϵ (not to be confused with the empty set $\emptyset$).
  - The textbook denotes the empty expression by $\emptyset$, but we will write ϵ instead, to prevent confusion.
- Two expressions U and V are equal iff they are of the same length and have the same symbols in the same order.
- Scanning of expressions proceeds from left to right.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.
- Every expression is a segment of itself.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.
- Every expression is a segment of itself.
- The empty expression ϵ is a segment of every expression.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.
- Every expression is a segment of itself.
- The empty expression ϵ is a segment of every expression.
- If $U = VW$, where U, V, W are expressions, then V is an initial segment (prefix) and W is a terminal segment (suffix) of U.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.
- Every expression is a segment of itself.
- The empty expression ϵ is a segment of every expression.
- If $U = VW$, where U, V, W are expressions, then V is an initial segment (prefix) and W is a terminal segment (suffix) of U.
  - If $W \neq \epsilon$, then V is a proper initial segment (proper prefix) of U.

Expressions

- Concatenating two expressions U, V, in this order, is denoted by UV. Note that $\epsilon U = U\epsilon = U$ for any expression U.
- If $U = W_1 V W_2$ where U, V, W_1, W_2 are expressions, then:
  - V is a segment of U,
  - If $V \neq U$, then V is a proper segment of U.
- Every expression is a segment of itself.
- The empty expression ϵ is a segment of every expression.
- If $U = VW$, where U, V, W are expressions, then V is an initial segment (prefix) and W is a terminal segment (suffix) of U.
  - If $W \neq \epsilon$, then V is a proper initial segment (proper prefix) of U.
  - If $V \neq \epsilon$, then W is a proper terminal segment (proper suffix) of U.

The set of formulas of $\mathcal{L}^p$

Definition. Atom$(\mathcal{L}^p)$ - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

The set of formulas of $\mathcal{L}^p$

Definition. Atom$(\mathcal{L}^p)$ - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

Definition:The set Form$(\mathcal{L}^p)$, of formulas of $\mathcal{L}^p$, is defined recursively as:

The set of formulas of $\mathcal{L}^p$

Definition. Atom($\mathcal{L}^p$) - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

Definition:The set Form($\mathcal{L}^p$), of formulas of $\mathcal{L}^p$, is defined recursively as:

BASE: Every atom in Atom($\mathcal{L}^p$) is a formula in Form($\mathcal{L}^p$);

The set of formulas of $\mathcal{L}^p$

Definition. Atom($\mathcal{L}^p$) - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

Definition:The set Form($\mathcal{L}^p$), of formulas of $\mathcal{L}^p$, is defined recursively as:

BASE: Every atom in Atom($\mathcal{L}^p$) is a formula in Form($\mathcal{L}^p$);

RECURSION: If A and B are formulas in Form($\mathcal{L}^p$), then:

1. $(\neg A)$ is a formula in Form($\mathcal{L}^p$),
2. $(A \wedge B)$ is a formula in Form($\mathcal{L}^p$),
3. $(A \vee B)$ is a formula in Form($\mathcal{L}^p$),
4. $(A \rightarrow B)$ is a formula in Form($\mathcal{L}^p$),
5. $(A \leftrightarrow B)$ is a formula in Form($\mathcal{L}^p$),

The set of formulas of $\mathcal{L}^p$

Definition. Atom$(\mathcal{L}^p)$ - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

Definition:The set Form$(\mathcal{L}^p)$, of formulas of $\mathcal{L}^p$, is defined recursively as:

BASE: Every atom in Atom$(\mathcal{L}^p)$ is a formula in Form$(\mathcal{L}^p)$;

RECURSION: If A and B are formulas in Form$(\mathcal{L}^p)$, then:

1. $(\neg A)$ is a formula in Form$(\mathcal{L}^p)$,
2. $(A \wedge B)$ is a formula in Form$(\mathcal{L}^p)$,
3. $(A \vee B)$ is a formula in Form$(\mathcal{L}^p)$,
4. $(A \rightarrow B)$ is a formula in Form$(\mathcal{L}^p)$,
5. $(A \leftrightarrow B)$ is a formula in Form$(\mathcal{L}^p)$,

RESTRICTION: No other expressions in $\mathcal{L}^p$ are formulas in Form$(\mathcal{L}^p)$.

The set of formulas of $\mathcal{L}^p$

Definition. Atom$(\mathcal{L}^p)$ - the atoms, or atomic formulas of $\mathcal{L}^p$ - is the set of expressions of $\mathcal{L}^p$ that consist of a proposition symbol only.

Definition:The set Form$(\mathcal{L}^p)$, of formulas of $\mathcal{L}^p$, is defined recursively as:

BASE: Every atom in Atom$(\mathcal{L}^p)$ is a formula in Form$(\mathcal{L}^p)$;

RECURSION: If A and B are formulas in Form$(\mathcal{L}^p)$, then:

1. $(\neg A)$ is a formula in Form$(\mathcal{L}^p)$,
2. $(A \wedge B)$ is a formula in Form$(\mathcal{L}^p)$,
3. $(A \vee B)$ is a formula in Form$(\mathcal{L}^p)$,
4. $(A \rightarrow B)$ is a formula in Form$(\mathcal{L}^p)$,
5. $(A \leftrightarrow B)$ is a formula in Form$(\mathcal{L}^p)$,

RESTRICTION: No other expressions in $\mathcal{L}^p$ are formulas in Form$(\mathcal{L}^p)$.

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.
- p, q, r are atomic formulas in Atom($\mathcal{L}^p$), and thus formulas in Form($\mathcal{L}^p$).

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.
- p, q, r are atomic formulas in Atom($\mathcal{L}^p$), and thus formulas in Form($\mathcal{L}^p$).
- $((p \wedge q) \rightarrow r)$ and $((\neg q) \leftrightarrow (p \vee s))$ are formulas in Form($\mathcal{L}^p$), but not atomic formulas in Atom($\mathcal{L}^p$).

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.
- p, q, r are atomic formulas in Atom($\mathcal{L}^p$), and thus formulas in Form($\mathcal{L}^p$).
- $((p \wedge q) \rightarrow r)$ and $((\neg q) \leftrightarrow (p \vee s))$ are formulas in Form($\mathcal{L}^p$), but not atomic formulas in Atom($\mathcal{L}^p$).
- $p \wedge \wedge \wedge (((r \rightarrow$ is an expression in $\mathcal{L}^p$, but it is neither an atomic formula in Atom($\mathcal{L}^p$), nor a formula in Form($\mathcal{L}^p$).

Comments and examples

- Items [1]- [5] in the RECURSION part of the definition of Form($\mathcal{L}^p$) are called the formation rules of formulas in $\mathcal{L}^p$.
- p, q, r are atomic formulas in Atom($\mathcal{L}^p$), and thus formulas in Form($\mathcal{L}^p$).
- $((p \wedge q) \rightarrow r)$ and $((\neg q) \leftrightarrow (p \vee s))$ are formulas in Form($\mathcal{L}^p$), but not atomic formulas in Atom($\mathcal{L}^p$).
- $p \wedge \wedge \wedge (((r \rightarrow$ is an expression in $\mathcal{L}^p$, but it is neither an atomic formula in Atom($\mathcal{L}^p$), nor a formula in Form($\mathcal{L}^p$).

1. Syntax of propositional language $\mathcal{L}^p$: atoms, formulas
2. **Generating and parsing formulas of $\mathcal{L}^p$**
3. Mathematical induction (optional), structural induction
4. Unique Readability Theorem
5. Precedence rules for connectives, scope of a connective

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

- p, q, r are in Form($\mathcal{L}^p$) by Definition of Form($\mathcal{L}^p$), BASE

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

- p, q, r are in Form($\mathcal{L}^p$) by Definition of Form($\mathcal{L}^p$), BASE
- $(\neg p)$ is in Form($\mathcal{L}^p$), by RECURSION rule [1]

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

- p, q, r are in Form($\mathcal{L}^p$) by Definition of Form($\mathcal{L}^p$), BASE
- $(\neg p)$ is in Form($\mathcal{L}^p$), by RECURSION rule [1]
- $(q \wedge r)$ and $(p \vee q)$ are in Form($\mathcal{L}^p$) due to RECURSION rules [2], respectively [3]

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

- p, q, r are in Form($\mathcal{L}^p$) by Definition of Form($\mathcal{L}^p$), BASE
- $(\neg p)$ is in Form($\mathcal{L}^p$), by RECURSION rule [1]
- $(q \wedge r)$ and $(p \vee q)$ are in Form($\mathcal{L}^p$) due to RECURSION rules [2], respectively [3]
- $((\neg p) \leftrightarrow (q \wedge r))$ is in Form($\mathcal{L}^p$), due to RECURSION rule [5] applied to $(\neg p)$ and $(q \wedge r)$

Generating formulas

The expression

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

is a formula. How is it generated using the formation rules?

- p, q, r are in Form($\mathcal{L}^p$) by Definition of Form($\mathcal{L}^p$), BASE
- $(\neg p)$ is in Form($\mathcal{L}^p$), by RECURSION rule [1]
- $(q \wedge r)$ and $(p \vee q)$ are in Form($\mathcal{L}^p$) due to RECURSION rules [2], respectively [3]
- $((\neg p) \leftrightarrow (q \wedge r))$ is in Form($\mathcal{L}^p$), due to RECURSION rule [5] applied to $(\neg p)$ and $(q \wedge r)$
- $((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$ is in Form($\mathcal{L}^p$), by RECURSION rule [4] applied to $(p \vee q)$ and $((\neg p) \leftrightarrow (q \wedge r))$

Parsing formulas

If Michelle wins at the Olympics, everyone will admire her, and she will get rich, but if she does not win, all her effort was in vain.

Parsing formulas

If Michelle wins at the Olympics, everyone will admire her, and she will get rich, but if she does not win, all her effort was in vain.

p: Michelle wins at the olympics.
q: Everyone admires Michelle.
r: Michelle will get rich.
s: Michelle's effort was in vain.

Parsing formulas

If Michelle wins at the Olympics, everyone will admire her, and she will get rich, but if she does not win, all her effort was in vain.

p: Michelle wins at the olympics.
q: Everyone admires Michelle.
r: Michelle will get rich.
s: Michelle's effort was in vain.

The compound proposition becomes

$$((p \rightarrow (q \wedge r)) \wedge ((\neg p) \rightarrow s))$$

One can use parse trees to analyze formulas.

Parse tree

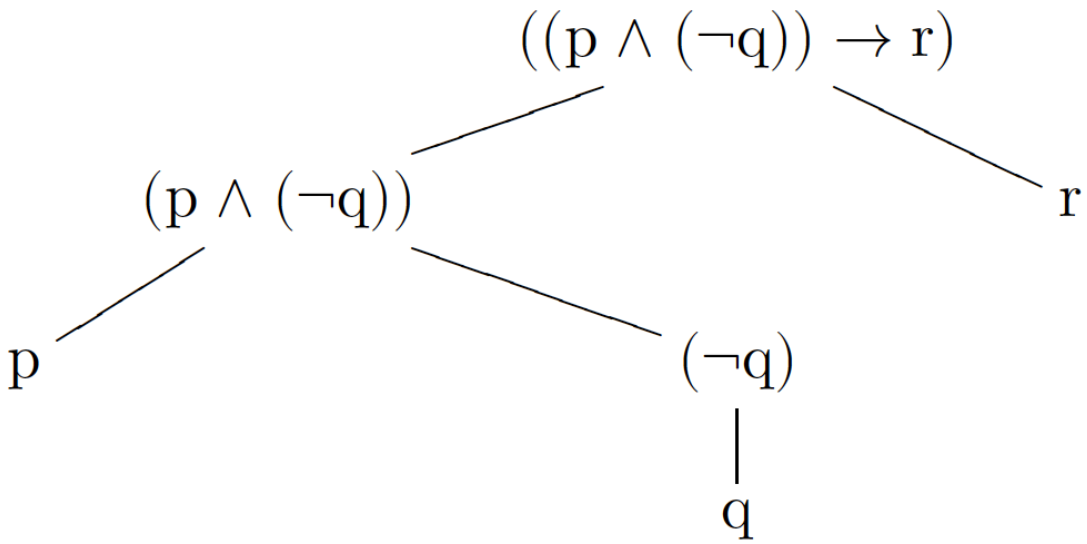
For example, this is a parse tree parse tree for the formula

$$((p \wedge (\neg q)) \rightarrow r)$$

Parse tree

For example, this is a parse tree parse tree for the formula

$$((p \wedge (\neg q)) \rightarrow r)$$



Note: Later in this module we will informally describe an algorithm that determines the main connective of a formula, i.e., the root of its parse tree (as a consequence of the Unique Readability Theorem).

Question: Can a formula be of two (or more) kinds?

Question: Can a formula be of two (or more) kinds?

For example, can it be both a conjunction and an implication?

Question: Can a formula be of two (or more) kinds?

For example, can it be both a conjunction and an implication?

Or both a negation and a disjunction?

Claims

- Every formula in Form$(\mathcal{L}^p)$ has the same number of occurrences of left and right parentheses.

Claims

- Every formula in Form$(\mathcal{L}^p)$ has the same number of occurrences of left and right parentheses.
- Any non-empty proper initial segment of a formula in Form$(\mathcal{L}^p)$ has more occurrences of left than right parentheses. Any non-empty proper terminal segment of a formula in Form$(\mathcal{L}^p)$ has fewer occurrences of left than right parentheses.

Claims

- Every formula in Form$(\mathcal{L}^p)$ has the same number of occurrences of left and right parentheses.
- Any non-empty proper initial segment of a formula in Form$(\mathcal{L}^p)$ has more occurrences of left than right parentheses. Any non-empty proper terminal segment of a formula in Form$(\mathcal{L}^p)$ has fewer occurrences of left than right parentheses.
- Neither a non-empty proper initial segment nor a non-empty proper terminal segment of a formula can itself be a formula in Form$(\mathcal{L}^p)$.

Claims

- Every formula in Form$(\mathcal{L}^p)$ has the same number of occurrences of left and right parentheses.
- Any non-empty proper initial segment of a formula in Form$(\mathcal{L}^p)$ has more occurrences of left than right parentheses. Any non-empty proper terminal segment of a formula in Form$(\mathcal{L}^p)$ has fewer occurrences of left than right parentheses.
- Neither a non-empty proper initial segment nor a non-empty proper terminal segment of a formula can itself be a formula in Form$(\mathcal{L}^p)$.
- (Unique Readability Theorem) Every formula in Form$(\mathcal{L}^p)$ is of exactly one of the six forms: an atom, $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, or $(A \leftrightarrow B)$ and, in each case, it is of that form in exactly one way.

Claims

- Every formula in Form($\mathcal{L}^p$) has the same number of occurrences of left and right parentheses.
- Any non-empty proper initial segment of a formula in Form($\mathcal{L}^p$) has more occurrences of left than right parentheses. Any non-empty proper terminal segment of a formula in Form($\mathcal{L}^p$) has fewer occurrences of left than right parentheses.
- Neither a non-empty proper initial segment nor a non-empty proper terminal segment of a formula can itself be a formula in Form($\mathcal{L}^p$).
- (Unique Readability Theorem) Every formula in Form($\mathcal{L}^p$) is of exactly one of the six forms: an atom, $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, or $(A \leftrightarrow B)$ and, in each case, it is of that form in exactly one way.
- To prove such claims, we will use induction.

1. Syntax of propositional language $\mathcal{L}^p$: atoms, formulas
2. Generating and parsing formulas of $\mathcal{L}^p$
3. **Mathematical induction (optional), structural induction**
4. Unique Readability Theorem
5. Precedence rules for connectives, scope of a connective

Review of mathematical induction

The "natural numbers" are the numbers we use to count things.
Before we start, we count 0; as we find things we count 1, 2, etc.
The natural numbers form an unbounded sequence

0, 1, 2, 3, 4, ...

Suppose P names a property. We write "$P(2)$" to mean "2 has property P", or "P holds for 2".

Review of mathematical induction

The "natural numbers" are the numbers we use to count things.
Before we start, we count 0; as we find things we count 1, 2, etc.
The natural numbers form an unbounded sequence

0, 1, 2, 3, 4, ...

Suppose P names a property. We write "$P(2)$" to mean "2 has property P", or "P holds for 2".

A statement "every natural number has property P" corresponds to a sequence of statements

$$P(0),\ P(1),\ P(2),\ P(3),\ P(4),\ \ldots$$

Mathematical induction

Principle of mathematical induction:

Suppose we establish two things:

1. 0 has property P, and
2. whenever a natural number has property P, then the next natural number also has property P.

Mathematical induction

Principle of mathematical induction:

Suppose we establish two things:

1. 0 has property P, and
2. whenever a natural number has property P, then the next natural number also has property P.

Then we may conclude that every natural number has property P.

Mathematical induction example

Example: Show that $\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$ for every natural number n.

Let P be the property; that is, let $P(n)$ be "$\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$".

Proof for the example

Step 1 (Base Case): The property $P(0)$ is $\sum_{x=0}^{0} x = \frac{0(0+1)}{2}$.

The left side of the equation is just 0. Also the right side evaluates to 0.
Thus 0 has property P.

Proof for the example

Step 1 (Base Case): The property $P(0)$ is $\sum_{x=0}^{0} x = \frac{0(0+1)}{2}$.

The left side of the equation is just 0. Also the right side evaluates to 0. Thus 0 has property P.

Step 2 (Inductive Step): Hypothesize that an arbitrary natural number, say k, has property P (this is called the Inductive Hypothesis, I.H.), that is,

$$\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$$

Note: We chose the letter k to differentiate this arbitrary but fixed natural number, from n which denotes any natural number.

Step 2 (Inductive Step), continued:

We hypothesized that k has property P, that is, $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

Step 2 (Inductive Step), continued:

We hypothesized that k has property P, that is, $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

We now need to demonstrate that $k+1$ has property P, that is,

$$\sum_{x=0}^{k+1} x = \frac{(k+1)\left((k+1)+1\right)}{2} = \frac{(k+1)\,(k+2)}{2}$$

Step 2 (Inductive Step), continued:

We hypothesized that k has property P, that is, $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

We now need to demonstrate that $k+1$ has property P, that is,

$$\sum_{x=0}^{k+1} x = \frac{(k+1)\left((k+1)+1\right)}{2} = \frac{(k+1)\,(k+2)}{2}$$

We calculate:

$$\begin{aligned}
\sum_{x=0}^{k+1} x &= \left(\sum_{x=0}^{k} x\right) + (k+1) && \text{definition of } \textstyle\sum \\
&= \tfrac{k(k+1)}{2} + (k+1) && \text{by the Inductive Hypothesis} \\
&= \left(\tfrac{k}{2} + 1\right)(k+1) && \text{"algebra"} \\
&= \tfrac{(k+1)(k+2)}{2} && \text{DONE!}
\end{aligned}$$

Observations/Techniques

To talk about something, give it a name.

E.g., property P, number k, etc.

Observations/Techniques

To talk about something, give it a name.

E.g., property P, number k, etc.

A formula is a textual object. In this text, we can substitute one symbol or expression for another. For example, we put "$k+1$" in place of "k".

Observations/Techniques

To talk about something, give it a name.

E.g., property P, number k, etc.

A formula is a textual object. In this text, we can substitute one symbol or expression for another. For example, we put "$k+1$" in place of "k".

The induction principle gives a "template" for a proof:

- The proof has two parts: "Base Case" and "Inductive Step".
- In the Inductive Step, hypothesize $P(k)$ and prove $P(k+1)$ from it.

Observations/Techniques

To talk about something, give it a name.

E.g., property P, number k, etc.

A formula is a textual object. In this text, we can substitute one symbol or expression for another. For example, we put "$k+1$" in place of "k".

The induction principle gives a "template" for a proof:

- The proof has two parts: "Base Case" and "Inductive Step".
- In the Inductive Step, hypothesize $P(k)$ and prove $P(k+1)$ from it.

The induction principle does not say how to actually do either step. We must invent those parts ourselves, and every problem is different.

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $P(0)$	Show $P(0)$
Ind. Hyp.:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $P(0)$	Show $P(0)$
Ind. Hyp.:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $P(0)$	Show $P(0)$
Ind. Hyp.:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- *$Q(0)$ is equivalent to $P(0)$.*

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $Q(0)$	Show $P(0)$
Ind. Hyp.:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- *$Q(0)$ is equivalent to $P(0)$.*
- *$Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".*
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $Q(0)$	Show $P(0)$
Ind. Hyp.:	$Q(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $Q(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- *$Q(0)$ is equivalent to $P(0)$.*
- *$Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".*
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.
- *"$Q(k)$ for every k" is equivalent to "$P(k)$ for every k".*

"Simple" induction vs. "Strong" induction

	Simple Induction	**Strong Induction** *(course-of-values induction)*
Base Case:	Show $Q(0)$	Show $P(0)$
Ind. Hyp.:	$Q(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $Q(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$Q(k)$ holds for every k	$P(k)$ holds for every k

What is the difference? No difference!

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- *$Q(0)$ is equivalent to $P(0)$.*
- *$Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".*
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.
- *"$Q(k)$ for every k" is equivalent to "$P(k)$ for every k".*

Induction example: Tiling checkerboards

Tiling Checkerboards

Example: Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes.

A right triomino is an L-shaped tile which covers three squares at a time.

Induction example: Tiling checkerboards

Tiling Checkerboards

Example: Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes.

A right triomino is an L-shaped tile which covers three squares at a time.

Solution: Let $P(n)$ be the proposition that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes. Use mathematical induction to prove that $P(n)$ is true for all positive integers n.

- BASIS STEP: P(1) is true, because each of the four 2×2 checkerboards with one square removed can be tiled using one right triomino.

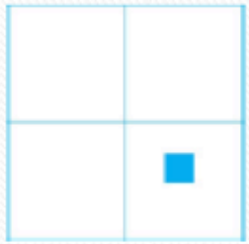
- INDUCTIVE STEP: Assume that $P(k)$ is true for every $2^k \times 2^k$ checkerboard, for some positive integer k.

Induction example: Tiling checkerboards

Tiling Checkerboards

Inductive Hypothesis: Every $2^k \times 2^k$ checkerboard, for some positive integer k, with one square removed can be tiled using right triominoes.

- Consider a $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed. Split this checkerboard into four checkerboards of size $2^k \times 2^k$, by dividing it in half in both directions.



- Remove a square from one of the four $2^k \times 2^k$ checkerboards. By the inductive hypothesis, this board can be tiled. Also by the inductive hypothesis, the other three boards can be tiled with the square from the corner of the center of the original board removed. We can then cover the three adjacent squares with a triominoe.
- Hence, the entire $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed can be tiled using right triominoes.

◀

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.
  Consider $k+1$ horses. Exclude the last horse and look only at the first k horses; all these are the same color by I.H.

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.
  Consider $k+1$ horses. Exclude the last horse and look only at the first k horses; all these are the same color by I.H.
  Likewise, exclude the first horse and look only at the last k horses. These too, must also be of the same color, by the I.H.

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.
  Consider $k+1$ horses. Exclude the last horse and look only at the first k horses; all these are the same color by I.H.
  Likewise, exclude the first horse and look only at the last k horses. These too, must also be of the same color, by the I.H.
  Therefore, the first horse in the group is of the same color as the horses in the middle, who in turn are of the same color as the last horse. Hence the first horse, middle horses, and last horse are all of the same color.

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.
  Consider $k+1$ horses. Exclude the last horse and look only at the first k horses; all these are the same color by I.H.
  Likewise, exclude the first horse and look only at the last k horses. These too, must also be of the same color, by the I.H.
  Therefore, the first horse in the group is of the same color as the horses in the middle, who in turn are of the same color as the last horse. Hence the first horse, middle horses, and last horse are all of the same color.
  What's wrong with this proof?

All horses are the same colour

We want to prove: "Any set S_n of n horses has the property $P(n)$: All horses in the set S_n are of the same colour, for every $n \geq 1$."

- Base Case: $P(1) =$ true - trivial
- Inductive Step: Assume $P(k)$ (I.H.) and prove $P(k+1)$.
  Consider $k+1$ horses. Exclude the last horse and look only at the first k horses; all these are the same color by I.H.
  Likewise, exclude the first horse and look only at the last k horses. These too, must also be of the same color, by the I.H.
  Therefore, the first horse in the group is of the same color as the horses in the middle, who in turn are of the same color as the last horse. Hence the first horse, middle horses, and last horse are all of the same color.
  What's wrong with this proof?
  The proof of the Inductive Step
  does not work for the Base Case, $k = 1$.

How to prove properties of formulas

We want to prove: "Every formula in Form$(\mathcal{L}^p)$ has property P".

A formula is not a natural number, but it suffices to prove any one of the following.

> *For every natural number n, every formula with n or fewer symbols has property P.*

OR

> *For every natural number n, every formula with n or fewer connectives has property P.*

OR

> *For every natural number n, every formula whose parse tree has height less than or equal to n has property P.*

OR

> *For every natural number n, every formula producible with n or fewer uses of the formation rules has property P.*

Recursively defined sets

Alternatively, we can use the fact that Form($\mathcal{L}^p$) is a recursively defined set, and use structural induction to prove properties about formulas in Form($\mathcal{L}^p$).

Recursively defined sets

Alternatively, we can use the fact that Form($\mathcal{L}^p$) is a recursively defined set, and use structural induction to prove properties about formulas in Form($\mathcal{L}^p$).

To define a set of objects recursively, we identify a few core objects as being in the set, and give rules showing how to build new set objects from the old. Formally, a recursive definition of a set consists of:

Recursively defined sets

Alternatively, we can use the fact that Form($\mathcal{L}^p$) is a recursively defined set, and use structural induction to prove properties about formulas in Form($\mathcal{L}^p$).

To define a set of objects recursively, we identify a few core objects as being in the set, and give rules showing how to build new set objects from the old. Formally, a recursive definition of a set consists of:

I. BASE: A statement that certain objects belong to the set.

Recursively defined sets

Alternatively, we can use the fact that Form($\mathcal{L}^p$) is a recursively defined set, and use structural induction to prove properties about formulas in Form($\mathcal{L}^p$).

To define a set of objects recursively, we identify a few core objects as being in the set, and give rules showing how to build new set objects from the old. Formally, a recursive definition of a set consists of:

I. BASE: A statement that certain objects belong to the set.

II. RECURSION: A collection of rules indicating how to form new set objects from those already known to be in the set.

Recursively defined sets

Alternatively, we can use the fact that Form($\mathcal{L}^p$) is a recursively defined set, and use structural induction to prove properties about formulas in Form($\mathcal{L}^p$).

To define a set of objects recursively, we identify a few core objects as being in the set, and give rules showing how to build new set objects from the old. Formally, a recursive definition of a set consists of:

I. BASE: A statement that certain objects belong to the set.

II. RECURSION: A collection of rules indicating how to form new set objects from those already known to be in the set.

III. RESTRICTION: A statement that no objects belong to the set other than those coming from I, and II.

Examples

Example: The set of natural numbers $\mathbb{N}$ is a recursively defined set with one formation rule ("add 1").

I. BASE: 0 is a natural number in $\mathbb{N}$.

II. RECURSION: If k is a natural number in $\mathbb{N}$, then $k+1$ is a natural number in $\mathbb{N}$.

III. RESTRICTION: No other numbers are in $\mathbb{N}$.

Examples

Example: The set of natural numbers $\mathbb{N}$ is a recursively defined set with one formation rule ("add 1").

I. BASE: 0 is a natural number in $\mathbb{N}$.

II. RECURSION: If k is a natural number in $\mathbb{N}$, then $k+1$ is a natural number in $\mathbb{N}$.

III. RESTRICTION: No other numbers are in $\mathbb{N}$.

Example: The set Form$(\mathcal{L}^p)$ is a recursively defined set, where the core set in BASE consist of all atomic formulas (proposition symbols), and the RECURSION has 5 formation rules (one for each connective).

Properties of recursively defined sets

Structural Induction for Recursively Defined Sets

Properties of recursively defined sets

Structural Induction for Recursively Defined Sets

Let S be a recursively defined set, and consider a property that objects in S may or may not have. The proof that every object in S satisfies the property consists of:

Properties of recursively defined sets

Structural Induction for Recursively Defined Sets

Let S be a recursively defined set, and consider a property that objects in S may or may not have. The proof that every object in S satisfies the property consists of:

(I) BASE CASE(S): Show that each object in the BASE for S satisfies the property;

Properties of recursively defined sets

Structural Induction for Recursively Defined Sets

Let S be a recursively defined set, and consider a property that objects in S may or may not have. The proof that every object in S satisfies the property consists of:

(I) BASE CASE(S): Show that each object in the BASE for S satisfies the property;

(II) (COMPOSITE) INDUCTIVE STEP: Show that, for **each** rule in the RECURSION of S, if the rule is applied to objects in S that satisfy the property, then the new objects defined by the rule also satisfy the property;

Properties of recursively defined sets

Structural Induction for Recursively Defined Sets

Let S be a recursively defined set, and consider a property that objects in S may or may not have. The proof that every object in S satisfies the property consists of:

(I) BASE CASE(S): Show that each object in the BASE for S satisfies the property;

(II) (COMPOSITE) INDUCTIVE STEP: Show that, for **each** rule in the RECURSION of S, if the rule is applied to objects in S that satisfy the property, then the new objects defined by the rule also satisfy the property;

Because, due to RESTRICTION, no objects other than those obtained through the BASE and RECURSION conditions are contained in S, it must then be the case than every object in S satisfies the property.

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

- Every atomic formula $p \in$ Atom$(\mathcal{L}^p)$ satisfies property R, and

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

- Every atomic formula $p \in$Atom$(\mathcal{L}^p)$ satisfies property R, and
- If formulas A and B in Form$(\mathcal{L}^p)$ satisfy property R, then:
  1. $(\neg A)$ satisfies property R,
  2. $(A \wedge B)$ satisfies has property R,
  3. $(A \vee B)$ satisfies has property R,
  4. $(A \rightarrow B)$ satisfies has property R,
  5. $(A \leftrightarrow B)$ satisfies has property R,

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

- Every atomic formula $p \in$ Atom$(\mathcal{L}^p)$ satisfies property R, and
- If formulas A and B in Form$(\mathcal{L}^p)$ satisfy property R, then:
  1. $(\neg A)$ satisfies property R,
  2. $(A \wedge B)$ satisfies has property R,
  3. $(A \vee B)$ satisfies has property R,
  4. $(A \rightarrow B)$ satisfies has property R,
  5. $(A \leftrightarrow B)$ satisfies has property R,

it follows that every formula in Form$(\mathcal{L}^p)$ satisfies property R.

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

- Every atomic formula $p \in$ Atom$(\mathcal{L}^p)$ satisfies property R, and
- If formulas A and B in Form$(\mathcal{L}^p)$ satisfy property R, then:
  1. $(\neg A)$ satisfies property R,
  2. $(A \wedge B)$ satisfies has property R,
  3. $(A \vee B)$ satisfies has property R,
  4. $(A \rightarrow B)$ satisfies has property R,
  5. $(A \leftrightarrow B)$ satisfies has property R,

it follows that every formula in Form$(\mathcal{L}^p)$ satisfies property R.

Note: If the set S is the set of natural numbers, structural induction amounts to classical mathematical induction.

Structural induction applied to Form$(\mathcal{L}^p)$

Theorem. Suppose R is a property. If

- Every atomic formula $p \in$Atom$(\mathcal{L}^p)$ satisfies property R, and
- If formulas A and B in Form$(\mathcal{L}^p)$ satisfy property R, then:
  1. $(\neg A)$ satisfies property R,
  2. $(A \wedge B)$ satisfies has property R,
  3. $(A \vee B)$ satisfies has property R,
  4. $(A \rightarrow B)$ satisfies has property R,
  5. $(A \leftrightarrow B)$ satisfies has property R,

it follows that every formula in Form$(\mathcal{L}^p)$ satisfies property R.

Note: If the set S is the set of natural numbers, structural induction amounts to classical mathematical induction.

Note: In general, the number of subcases of the composite inductive step in a proof by structural induction equals the number of formation rules in the RECURSION part of the definition of that set (e.g., 1 rule/subcase for $\mathbb{N}$, and 5 rules/subcases for Form$(\mathcal{L}^p)$).

Example: Parentheses in formulas

To illustrate structural induction, we shall prove the following.

Lemma 1. *Every formula in* Form$(\mathcal{L}^p)$ *has an equal number of left and right parentheses.*

Example: Parentheses in formulas

To illustrate structural induction, we shall prove the following.

> *Lemma 1. Every formula in* Form$(\mathcal{L}^p)$ *has an equal number of left and right parentheses.*

Proof. We use structural induction. The property to prove is

> $R(A)$: A *has an equal number of left and right parentheses*

for every formula A in Form$(\mathcal{L}^p)$.

Example: Parentheses in formulas

To illustrate structural induction, we shall prove the following.

> *Lemma 1. Every formula in* Form$(\mathcal{L}^p)$ *has an equal number of left and right parentheses.*

Proof. We use structural induction. The property to prove is

> $R(A)$: A *has an equal number of left and right parentheses*

for every formula A in Form$(\mathcal{L}^p)$.

Base Case: A is an atom. Then, A has zero left and right parentheses, as it is only a proposition symbol. Thus $R(A)$ holds. This completes the proof of the Base Case.

Proof cont'd: Inductive Step, subcase $\neg$

Inductive Step, the subcase of $\neg$

Notation: For any formula A, let $\ell(A)$ denote the number of '(' in A, and let $r(A)$ denote the number of ')' in A.

Assume A is $(\neg B)$.

Proof cont'd: Inductive Step, subcase $\neg$

Inductive Step, the subcase of $\neg$

Notation: For any formula A, let $\ell(A)$ denote the number of '(' in A, and let $r(A)$ denote the number of ')' in A.

Assume A is $(\neg B)$.

Proof cont'd: Inductive Step, subcase $\neg$

Inductive Step, the subcase of $\neg$

Notation: For any formula A, let $\ell(A)$ denote the number of '(' in A, and let $r(A)$ denote the number of ')' in A.

Assume A is $(\neg B)$.

Inductive Hypothesis: formula B has property R, i.e. $\ell(B) = r(B)$.

Proof cont'd: Inductive Step, subcase $\neg$

Inductive Step, the subcase of $\neg$

Notation: For any formula A, let $\ell(A)$ denote the number of '(' in A, and let $r(A)$ denote the number of ')' in A.

Assume A is $(\neg B)$.

Inductive Hypothesis: formula B has property R, i.e. $\ell(B) = r(B)$.

Then we have

$$\begin{aligned} \ell\left((\neg B)\right) &= 1 + \ell(B) \text{ (inspection)} \\ &= 1 + r(B) \text{ (induction hypothesis: } R(B)) \\ &= r\left((\neg B)\right) \text{ (inspection)} \end{aligned}$$

Proof cont'd: Inductive Step, subcases $\wedge, \vee, \rightarrow, \leftrightarrow$

Inductive Hypothesis: formulas B and C both have property R.
To prove: Each of the formulas $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ and $(B \leftrightarrow C)$ has property R.

Proof cont'd: Inductive Step, subcases $\wedge, \vee, \rightarrow, \leftrightarrow$

Inductive Hypothesis: formulas B and C both have property R.
To prove: Each of the formulas $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ and $(B \leftrightarrow C)$ has property R.

Without loss of generality (w.l.o.g.), we consider $(B \wedge C)$.

Proof cont'd: Inductive Step, subcases $\wedge, \vee, \rightarrow, \leftrightarrow$

Inductive Hypothesis: formulas B and C both have property R.
To prove: Each of the formulas $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ and $(B \leftrightarrow C)$ has property R.

Without loss of generality (w.l.o.g.), we consider $(B \wedge C)$.
We calculate $\ell\big((B \wedge C)\big)$:

$$\begin{aligned} \ell\big((B \wedge C)\big) &= 1 + \ell(B) + \ell(C) && \text{inspection} \\ &= 1 + r(B) + r(C) && (I.H.)\ \ R(B) \text{ and } R(C) \end{aligned}$$

This concludes the proof of the (composite) inductive step, the inductive proof, and thus the Example.

Proof cont'd: Inductive Step, subcases $\wedge, \vee, \rightarrow, \leftrightarrow$

Inductive Hypothesis: formulas B and C both have property R.
To prove: Each of the formulas $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ and $(B \leftrightarrow C)$ has property R.

Without loss of generality (w.l.o.g.), we consider $(B \wedge C)$.
We calculate $\ell\big((B \wedge C)\big)$:

$$\begin{aligned} \ell\big((B \wedge C)\big) &= 1 + \ell(B) + \ell(C) && \text{inspection} \\ &= 1 + r(B) + r(C) && (I.H.)\ \ R(B) \text{ and } R(C) \\ &= r\big((B \wedge C)\big) && \text{inspection} \end{aligned}$$

This concludes the proof of the (composite) inductive step, the inductive proof, and thus the Example.

1. Syntax of propositional language $\mathcal{L}^p$: atoms, formulas
2. Generating and parsing formulas of $\mathcal{L}^p$
3. Mathematical induction (optional), structural induction
4. **Unique Readability Theorem**
5. Precedence rules for connectives, scope of a connective

The Unique Readability Theorem

Theorem (Unique Readability). *Every formula in* Form$(\mathcal{L}^p)$ *is of exactly one of the six forms: an atom,* $(\neg B)$, $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ *or* $(B \leftrightarrow C)$ *and, in each case, it is of that form in exactly one way.*

The Unique Readability Theorem

Theorem (Unique Readability). *Every formula in* Form$(\mathcal{L}^p)$ *is of exactly one of the six forms: an atom,* $(\neg B)$, $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$ *or* $(B \leftrightarrow C)$ *and, in each case, it is of that form in exactly one way.*

Proof. Use the following Claims:

(a) *The first symbol of a formula* A *is either '(' or a proposition symbol* (by the recursive definition of formulas)

(b) *Every formula* A *has an equal number of '(' and ')'* (*Lemma 1*)

(c) *If* A' *is a non-empty proper initial segment of a formula* A*, then* A' *has more '(' than ')'* (can be proved by structural induction)

Proof of the Unique Readability Theorem

The fact that every formula is of one of the six forms follows from the recursive definition of the set Form$(\mathcal{L}^p)$ of formulas.

Proof of the Unique Readability Theorem

The fact that every formula is of one of the six forms follows from the recursive definition of the set Form$(\mathcal{L}^p)$ of formulas.

Assume that a formula F can be of two different forms, that is:

1. $F = p = A$, where A is not an atom.
2. $F = (\neg A) = (B \star C)$, where $\star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
3. $F = (B \star C) = (B' \star' C')$ where $\star, \star' \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

where all of F, A, B, C, B', C' are formulas in Form$(\mathcal{L}^p)$.

Proof of the Unique Readability Theorem

The fact that every formula is of one of the six forms follows from the recursive definition of the set Form$(\mathcal{L}^p)$ of formulas.

Assume that a formula F can be of two different forms, that is:

1. $F = p = A$, where A is not an atom.
2. $F = (\neg A) = (B \star C)$, where $\star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
3. $F = (B \star C) = (B' \star' C')$ where $\star, \star' \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

where all of F, A, B, C, B', C' are formulas in Form$(\mathcal{L}^p)$.

Note: In, e.g., case (3), $(B \star C)$ and $(B' \star' C')$ are two decompositions of the same formula F, into its constituent (sub)formulas.

Recall that $(B \star C) = (B' \star' C')$ means that the formulas $(B \star C)$ and $(B' \star' C')$ are equal (identical), i.e., they have same length, and the same sequence of symbols, in the same order.

Proof of Unique Readability, cont'd

1. Assume $p = A$. Due to Claim (a), a non-atomic formula A starts with $'('$. Since the first two symbols on each side are the same, this implies $p \neq ($ which is a contradiction.
2. Assume $(\neg A) = (B \star C)$. Delete the first symbol on each side, obtaining $\neg A) = B \star C)$. This implies either $\neg = p$ (if $B = p$ is an atom), or $\neg = ($ (if B is a non-atomic formula), due to Claim (a), a contradiction.
3. Assume $(B \star C) = (B' \star' C')$. Delete the first symbol on each side, obtaining $B \star C) = B' \star' C')$.

   We now have three possibilities:

   (i) B has the same length as B'
   (ii) B is strictly shorter than B'
   (iii) B is strictly longer than B'

Proof of Unique Readability, cont'd

3(i) If B has the same length as B', then $B = B'$, $\star = \star'$, and $C = C'$.

3(ii) If B is strictly shorter than B', this implies that B is a non-empty initial segment of the formula B'. By Claim (c), this implies that B has more $'('$ than $')'$.
On the other hand, since B is itself a formula, by Claim (b), B has the same number of $'('$ and $')'$.
This is a contradiction.

3(iii) If B is strictly longer than B', reasoning similar to 3(ii) leads to a contradiction.

Thus, the only case that did not lead to a contradiction, 3(i), implies that the decomposition of a formula F into its constituent (sub)formulas is unique. □

Consequences of Unique Readability

There are six kinds of formulas:

- A proposition symbol p is called an atom.
- $(\neg A)$ is called a negation (formula). It is the negation of A.
- $(A \wedge B)$ is called a conjunction (formula). It is the conjunction of A and B. A and B are called the conjuncts of $(A \wedge B)$.
- $(A \vee B)$ is called a disjunction (formula). It is the (inclusive) disjunction of A and B. A and B are called the disjuncts of $(A \vee B)$.
- $(A \rightarrow B)$ is called an implication (formula). It is the implication of A and B. A and B are called the antecedent and consequent of $(A \rightarrow B)$.
- $(A \leftrightarrow B)$ is called an equivalence (formula). It is the equivalence of A and B.

Moreover, each formula is of exactly one kind.

Algorithms for parsing formulas

Algorithms for parsing formulas

The Unique Readability Theorem ensures unambiguous formulas.

Algorithms for parsing formulas

The Unique Readability Theorem ensures unambiguous formulas.

Algorithm: Given a formula in Form($\mathcal{L}^p$), determine its subformulas:

1	2	$\ldots$	m	$m+1$	$m+2$	$\ldots$	$n-1$	n
(	$\underbrace{(\ \langle\texttt{rest of subformula}\rangle\ )}$			$\star$	$\langle\texttt{subformula 2}\rangle$			**)**

To determine m: count excess of '(' over ')'

Algorithms for parsing formulas

The Unique Readability Theorem ensures unambiguous formulas.

Algorithm: Given a formula in Form($\mathcal{L}^p$), determine its subformulas:

1	2	$\ldots$	m	$m+1$	$m+2$	$\ldots$	$n-1$	n
(	$\underbrace{(\ \langle\texttt{rest of subformula}\rangle\ )}$			$\star$	$\langle\texttt{subformula 2}\rangle$			**)**

To determine m: count excess of '(' over ')'

After scanning the first left parenthesis, start a counter equal to the difference between the encountered number of left parentheses and the number of encountered right parentheses.

Algorithms for parsing formulas

The Unique Readability Theorem ensures unambiguous formulas.

Algorithm: Given a formula in Form($\mathcal{L}^p$), determine its subformulas:

1	2	$\ldots$	m	$m+1$	$m+2$	$\ldots$	$n-1$	n
(	$\underbrace{(\ \langle\texttt{rest of subformula}\rangle\ )}$			$\star$	$\langle\texttt{subformula 2}\rangle$			**)**

To determine m: count excess of '(' over ')'

After scanning the first left parenthesis, start a counter equal to the difference between the encountered number of left parentheses and the number of encountered right parentheses.
When the count returns zero, the first subformula (which starts at symbol 2 and ends at symbol m) has been determined.

1. Syntax of propositional language $\mathcal{L}^p$: atoms, formulas
2. Generating and parsing formulas of $\mathcal{L}^p$
3. Mathematical induction (optional), structural induction
4. Unique Readability Theorem
5. **Precedence rules for connectives, scope of a connective**

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$

$\wedge$ has precedence over $\vee$

$\vee$ has precedence over $\rightarrow$

$\rightarrow$ has precedence over $\leftrightarrow$

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$

$\wedge$ has precedence over $\vee$

$\vee$ has precedence over $\rightarrow$

$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as $((p \wedge q) \vee r)$.
- $p \rightarrow q \vee r$ is to be understood as

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as $((p \wedge q) \vee r)$.
- $p \rightarrow q \vee r$ is to be understood as $(p \rightarrow (q \vee r))$

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as $((p \wedge q) \vee r)$.
- $p \rightarrow q \vee r$ is to be understood as $(p \rightarrow (q \vee r))$
- $p \leftrightarrow p \rightarrow q$ must be understood as

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as $((p \wedge q) \vee r)$.
- $p \rightarrow q \vee r$ is to be understood as $(p \rightarrow (q \vee r))$
- $p \leftrightarrow p \rightarrow q$ must be understood as $(p \leftrightarrow (p \rightarrow q))$

Precedence rules (for human logicians)

$\neg$ has precedence over $\wedge$
$\wedge$ has precedence over $\vee$
$\vee$ has precedence over $\rightarrow$
$\rightarrow$ has precedence over $\leftrightarrow$

Examples:

- $\neg p \vee q$ is to be understood as $((\neg p) \vee q)$.
- $p \wedge q \vee r$ is to be understood as $((p \wedge q) \vee r)$.
- $p \rightarrow q \vee r$ is to be understood as $(p \rightarrow (q \vee r))$
- $p \leftrightarrow p \rightarrow q$ must be understood as $(p \leftrightarrow (p \rightarrow q))$
- The proposition in the example of parsing formulas can now be written without all the parentheses, as $(p \rightarrow q \wedge r) \wedge (\neg p \rightarrow s)$.

Scope

- If $(\neg A)$ is a segment of a formula C, then A is called the scope of this negation in the formula C.
- If $(A \wedge B)$ is a segment of a formula C, then A is called the left scope of the conjunction and B is called the right scope of this conjunction in the formula C.
- Similar definitions hold for left/right scopes of disjunction, implication and equivalence.

Example

Suppose $A = (\neg((p \wedge q) \vee ((\neg p) \rightarrow r)))$.

Example

Suppose $A = (\neg((p \wedge q) \vee ((\neg p) \rightarrow r)))$.

The scope of the first $\neg$ is $((p \wedge q) \vee ((\neg p) \rightarrow r))$ and of the second $\neg$ is p.

Example

Suppose $A = (\neg((p \wedge q) \vee ((\neg p) \rightarrow r)))$.

The scope of the first $\neg$ is $((p \wedge q) \vee ((\neg p) \rightarrow r))$ and of the second $\neg$ is p. The left and right scopes of $\wedge$ are p and q.

Example

Suppose $A = (\neg((p \wedge q) \vee ((\neg p) \rightarrow r)))$.

The scope of the first $\neg$ is $((p \wedge q) \vee ((\neg p) \rightarrow r))$ and of the second $\neg$ is p.
The left and right scopes of $\wedge$ are p and q.
The left and right scopes of $\vee$ are $(p \wedge q)$ and $((\neg p) \rightarrow r)$.

Example

Suppose $A = (\neg((p \wedge q) \vee ((\neg p) \rightarrow r)))$.

The scope of the first $\neg$ is $((p \wedge q) \vee ((\neg p) \rightarrow r))$ and of the second $\neg$ is p.
The left and right scopes of $\wedge$ are p and q.
The left and right scopes of $\vee$ are $(p \wedge q)$ and $((\neg p) \rightarrow r)$.
The left and right scopes of $\rightarrow$ are $(\neg p)$ and r.

Learning Objectives

- Know what is the propositional language $\mathcal{L}^p$ and the definitions of: expressions of $\mathcal{L}^p$, atoms (atomic formulas) of $\mathcal{L}^p$ (Atom($\mathcal{L}^p$)), and the set of formulas of $\mathcal{L}^p$ (Form($\mathcal{L}^p$))
- Know how to generate and parse formulas of $\mathcal{L}^p$ (parse tree)
- Review the mathematical induction proof technique
- Know the definition of a recursive set
- Know how to prove statements by structural induction
- Know the statement and proof of Unique Readability Theorem, and its consequences
- Know the precedence rules for logical connectives
- Know the definition of the scope of a connective