

Formal Proofs

Martin Fränzle

Informatics and Mathematical Modelling
The Technical University of Denmark

Junctors & Quantifiers

Some sample statements

Let $\text{father}(x, y)$ be the x -is-father-of- y predicate.

1. $\text{father}(x, y)$ applies to all **valuations** of x and y with x father of y ,
2. $\exists x.\text{father}(x, y)$ applies to all humans (except perhaps Adam and Eve),
3. $\exists y.\text{father}(x, y)$ applies to all male humans who have children,
4. $\forall y.\exists x.\text{father}(x, y)$ is valid (if we believe in evolution and refute creationism),
5. $\exists x.\forall y.\text{father}(x, y)$ is certainly not true (though some try hard to spread their genes...),
6. $\text{male}(x) \wedge \text{female}(x)$ is contradictory, i.e. not true of any x ,
7. $(\exists x.\text{male}(x)) \wedge (\exists x.\text{female}(x))$ is valid,
8. $(\exists x.(\text{male}(x) \wedge \text{female}(x)))$ is invalid,
9. $(\forall x.(\text{female}(x) \implies \neg \exists y \text{father}(x, y)))$ is valid.

02140 Languages and Parsing, MF, Fall 2004 – p.3/31

True statements

02140 Languages and Parsing, MF, Fall 2004 – p.4/31

Tautologies

If we claim a **statement** like

$$\text{if } x \geq 4 \text{ then } x^2 \leq 2^x$$

to be **true**, what do we mean?

then we claim that **it holds for all possible values of its “free” variables** (e.g., x).

A statement holding for all valuations of its variables is called a **tautology**.

Explicating hidden quantifiers

We can make this **implicit** reference to truth under arbitrary valuations explicit through quantifier introduction:

- if $x \geq 4$ then $x^2 \leq 2^x$
- if x is a real number and $x \geq 4$ then $x^2 \leq 2^x$
- $(x \in \mathbf{R} \wedge x \geq 4) \implies x^2 \leq 2^x$
- $\forall x. ((x \in \mathbf{R} \wedge x \geq 4) \implies x^2 \leq 2^x)$
- $\forall x \in \mathbf{R}. (x \geq 4 \implies x^2 \leq 2^x)$

Hiding explicit quantifiers

Within proofs, the converse operation of making quantifiers implicit is often useful:

- $\forall x.\phi$ $\rightsquigarrow$ Let x be **arbitrarily** given.
Then ϕ holds because ...
- $\forall x \in M.\phi$ $\rightsquigarrow$ Let x be an **arbitrary element of M** .
Then ϕ holds because ...
- $\exists x.\phi$ $\rightsquigarrow$ We **define** x by ...
Then ϕ holds because ...
- $\exists x \in M.\phi$ $\rightsquigarrow$ We **define** x by ...
Then x is in M because ...
and ϕ holds because ...

N.B.: $\forall$: unconstrained choice; $\exists$: explicit construction of witness!

02140 Languages and Parsing, MF, Fall 2004 – p.7/31

Example

Show

if $x \in \mathbb{Q} \setminus \{0\}$ then there is $y \in \mathbb{Q}$ with $xy = 1$.

which transliterates to

$$\forall x \in \mathbb{Q} \setminus \{0\}.\exists y \in \mathbb{Q}.xy = 1$$

Proof: Let $x \in \mathbb{Q} \setminus \{0\}$ be arbitrary.

We take $y \stackrel{\text{def}}{=} \frac{1}{x}$, which exists in $\mathbb{Q}$ because $\mathbb{Q} \setminus \{0\}$ is closed under division. Then $xy = 1$ holds, as $xy = x\frac{1}{x} = \frac{x}{x} = 1$.

Note that x is left unconstrained while y is constructed with reference to the particular choice of x , thus reflecting the quantifier nesting.

02140 Languages and Parsing, MF, Fall 2004 – p.8/31

Biimplications/iff

Often, you will encounter statements of the form

- ϕ holds if and only if ψ holds, or
- briefly, ϕ iff ψ , or
- formally, $\phi \iff \psi$.

These **biimplications** transliterate to

1. if ϕ holds then ψ holds, and
2. if ψ holds then ϕ holds.

ϕ is sufficient for ψ

ϕ is necessary for ψ

Normally, you prove the biimplication by proving the implications individually.

Example

Let $m, n \in \mathbf{N}$.

Conjecture: $m \leq n$ iff $\exists l \in \mathbf{N}. m + l = n$.

Proof:

1. We show that if $m \leq n$ then $\exists l \in \mathbf{N}. m + l = n$.

⋮

2. We show that if $\exists l \in \mathbf{N}. m + l = n$ then $m \leq n$.

⋮

Set equality

Set equality

- Two sets are equal iff they contain the same elements.
- $M = N$ iff $\forall x. (x \in M \iff x \in N)$

Set equality $M = N$ can be proven by

1. showing “if $x \in M$ then $x \in N$ ”, and
2. showing “if $x \in N$ then $x \in M$ ”.

Example

Conjecture: $\mathbb{Q} \setminus \{0\} = \left\{ \frac{1}{y} \mid y \in \mathbb{Q} \setminus \{0\} \right\}$.

Proof:

1. We show that if $x \in \mathbb{Q} \setminus \{0\}$ then $x \in \left\{ \frac{1}{y} \mid y \in \mathbb{Q} \setminus \{0\} \right\}$.

$\vdots$

2. We show that if $x \in \left\{ \frac{1}{y} \mid y \in \mathbb{Q} \setminus \{0\} \right\}$ then $x \in \mathbb{Q} \setminus \{0\}$.

$\vdots$

Proof by induction

Inductive proof

is a technique for showing properties $\phi(x)$ for each element $x \in M$ of a certain set M by

1. showing some base cases, and
2. using a recursive form of argument for the rest of the set M .

You may/should know induction on the natural numbers, where $M = \mathbb{N}$.

Induction applies to a much wider class of structures, including strings, lists, trees, ...

02140 Languages and Parsing, MF, Fall 2004 – p.15/31

Relations/orders

- A **binary relation** on a set A is a set $R \subseteq A \times A$.
- A binary relation R is a **(partial) order** iff
 1. R is **transitive**, i.e.

$$aRb \wedge bRc \implies aRc$$

2. R is **antisymmetric**, i.e.

$$aRb \wedge bRa \implies a = b$$

- An order is **total** iff

$$a \neq b \implies aRb \vee bRa$$

02140 Languages and Parsing, MF, Fall 2004 – p.16/31

Examples of partial orders

- $(\mathbf{N}, <)$, i.e. natural numbers with their strict native order: total,
- $(\mathbf{N}, \leq)$, i.e. natural numbers with their non-strict native order: total,
- $(\mathbf{N}, |)$, i.e. natural numbers with “divides” used as order: partial,
- $(\{1, 2, 4, 8, 256\}, |)$, i.e. some powers of 2 ordered by “divides”: total,
- $(\text{Strings}, \prec)$, i.e. strings with prefix order: partial,
- $(\text{Strings}, \prec_L)$, i.e. strings with lexicographic order: total,
- $(\text{Trees}, \prec)$, i.e. trees with subtree order: partial.

02140 Languages and Parsing, MF, Fall 2004 – p.17/31

Minimal elements

- An element $m \in A$ is called **minimal** wrt. a binary relation R on A iff

$$\neg (\exists b \in A \setminus \{m\}. b R m)$$

- Sets may have no, one, or multiple minimal elements wrt. a relation:
 - $\mathbf{Z}$ has no minimal element wrt. $<$,
 - $\mathbf{N}$ has one minimal element wrt. $<$,
 - $\{a, aa, bb, bba\}$ has two minimal elements wrt. $\prec$, namely a and bb ,
 - $\mathcal{P}(\mathbf{N})$ has one minimal element wrt. $\subseteq$, namely $\emptyset$.

02140 Languages and Parsing, MF, Fall 2004 – p.18/31

Well-founded relations

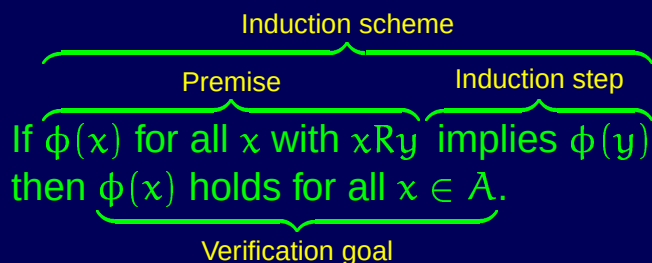
- A binary relation on A is called **well-founded** if every **nonempty** subset of A has at least one minimal element.
- Examples:
 - $<$ is **well-founded** on $\mathbb{N}$,
 - $<$ is **not well-founded** on $\mathbb{Z}$,
 - $\prec$ is **well-founded** on strings,
 - $\prec_L$ is **well-founded** on strings,
 - $\prec$ is **well-founded** on trees.

02140 Languages and Parsing, MF, Fall 2004 – p.19/31

Well-founded induction

Let $R \subseteq A \times A$ be a well-founded irreflexive relation.

Thm: (Principle of well-founded induction)



Proof: Assume the induction scheme is satisfied, yet $E \stackrel{\text{def}}{=} \{x \in A \mid \neg\phi(x)\} \neq \emptyset$.
Due to well-foundedness of R , there is a minimal element $e \in E$. This yields:

1. $\neg\phi(e)$, as $e \in E$.
2. $\phi(e)$, as
 - (a) xRe implies $x \notin E$ due to minimality of e in E , and $x \notin E$ implies $\phi(x)$,
 - (b) $\phi(x)$ for all x with xRe implies $\phi(e)$ due to the induction scheme.

This yields a contradiction, which means that at least part of the assumption cannot hold.

02140 Languages and Parsing, MF, Fall 2004 – p.20/31

Specializations

Well-founded induction is rarely applied directly. Instead, stronger verification conditions are used which imply the induction scheme of well-founded induction. The most frequent specialization used is that

1. some natural well-founded **order** on the set A is used as the relation R ,
2. the induction scheme is split in
 - (a) a so-called **base case** dealing with the minimal elements y of A , for which the set $\{x \in A \mid xRy\}$ is empty,
 - (b) an **induction step** applying to non-minimal elements and selecting some specific x 's with xRy instead of the full set $\{x \in A \mid xRy\}$.

02140 Languages and Parsing, MF, Fall 2004 – p.21/31

Specialization 1: Complete induction on $\mathbb{N}$

To show $\phi(n)$ for all $n \in \mathbb{N}$, prove

1. $\phi(0)$,
2. $\phi(\underbrace{k}_{< x})$ implies $\phi(\underbrace{k+1}_x)$ for arbitrary $k \in \mathbb{N}$

Base case

Induction step

Example: Show that $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ for all $n \in \mathbb{N}$.

02140 Languages and Parsing, MF, Fall 2004 – p.22/31

Specialization 2: Std. induction on strings/lists

To show $\phi(s)$ for all strings/lists s , prove

1. $\phi(\underbrace{\epsilon}_{\text{empty string/list}})$

Base case

2. $\phi(\underbrace{\langle s_1, \dots, s_k \rangle}_{\prec x}) \text{ implies } \phi(\underbrace{\langle s_1, \dots, s_k, s_{k+1} \rangle}_x)$

Ind. step

Example: Show that $\sum_{i=1}^n 2^{i-1} x_i < 2^n$ for each list $\langle x_1, \dots, x_n \rangle$ over $\{0, 1\}$.

02140 Languages and Parsing, MF, Fall 2004 – p.23/31

Specialization 3: Std. induction on trees

To show $\phi(t)$ for all trees t , prove

1. $\phi(\underbrace{\alpha}_{\text{minimal wrt. } \prec})$ for all atomic trees α ,

Base case

2. $\left(\forall t' \text{ direct subtree of } t. \phi(\underbrace{t'}_{\prec x}) \right) \text{ implies } \phi(\underbrace{t}_x)$

Ind. step

Atomic trees are the trees of depth 0.

Example: Show that a tree of depth n has at most 2^n leafs.

02140 Languages and Parsing, MF, Fall 2004 – p.24/31

Preliminaries for Formal Languages:

Alphabets, strings, languages

02140 Languages and Parsing, MF, Fall 2004 – p.25/31

What you'll learn

You will be introduced to the basic notions of

Alphabet: A set of “symbols”,

String/word: a sequence of symbols from the alphabet,

Language: a set of strings.

02140 Languages and Parsing, MF, Fall 2004 – p.26/31

Alphabets

Def.: An alphabet is a *finite, nonempty* set.

- Ex.:**
- The binary alphabet $\{0, 1\}$,
 - the set of all printable ASCII characters,
 - the set $\{\text{red, green, blue, yellow, magenta, cyan}\}$ of basic colors.

N.B.: This notion of alphabet is distinctly different from everyday use:

- alphabets need not feature an ordering,
- the members of an alphabet need not be letters in the usual sense, but can well be compounds of some kind (e.g., tuples).

We will often denote alphabets by the greek letter Σ .

02140 Languages and Parsing, MF, Fall 2004 – p.27/31

Strings (a.k.a. words)

Def.: A string (or word) over an Alphabet Σ is a *finite* sequence of symbols from Σ .

The length of a string w , denoted $|w|$, is the number of positions for symbols in the string.

- Ex.:**
- 0100 (or, more formally, $\langle 0, 1, 0, 0 \rangle$) is a string of length 4 over alphabet $\{0, 1\}$.
 - $\langle \rangle$ is the string of length 0 over alphabet $\{a, \dots, z\}$ (as well as over any other alphabet). It is called the *empty string* and is generically denoted ε , irrespective of the particular alphabet.

Def.: Given strings $v = \langle v_1, v_2, \dots, v_n \rangle$ and $w = \langle w_1, w_2, \dots, w_m \rangle$, the concatenation of v and w is the string

$$vw \stackrel{\text{def}}{=} \langle v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_m \rangle$$

of length $n + m$.

02140 Languages and Parsing, MF, Fall 2004 – p.28/31

Powers of an alphabet

Given an alphabet Σ and a natural number $n \in \mathbf{N}$,

- Σ^n is the set of all strings over Σ of length n ,
- $\Sigma^+ \stackrel{\text{def}}{=} \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{i=1}^{\infty} \Sigma^i$ is the set of all *nonempty* strings over Σ ,
- $\Sigma^* \stackrel{\text{def}}{=} \{\epsilon\} \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \{\epsilon\} \cup \Sigma^+$ is the set of all strings over Σ .

02140 Languages and Parsing, MF, Fall 2004 – p.29/31

Languages

Def.: A set of strings over an alphabet Σ (i.e., a subset of Σ^*) is called a language over Σ .

Ex.: • The set of strings consisting of n 0's followed by n 1's for arbitrary $n \in \mathbf{N}$ forms a language over $\{0, 1\}$:

$$\{\epsilon, 01, 0011, 000111, 00001111, \dots\} = \{0^n 1^n \mid n \in \mathbf{N}\}$$

- The set of strings over $\{0, \dots, 9\}$ representing a prime number forms a language over $\{0, \dots, 9\}$:

$$\{2, 3, 5, 7, 11, 13, 17, \dots\}$$

- Σ^* is a language over Σ ,
- $\emptyset$ is a language (over arbitrary alphabet),
- $\{\epsilon\}$ is a language (over arbitrary alphabet).

Note: $\{\epsilon\} \neq \emptyset$.

Note: Languages can be finite or infinite!

02140 Languages and Parsing, MF, Fall 2004 – p.30/31

Problems

Def.: Given a language L over some alphabet Σ , the problem L is:
Given a string w over Σ , decide whether or not $w \in L$.

- Ex.:**
- Problem of deciding whether some $w \in \{0,1\}^*$ is a binary representation of a prime number.
 - Problem of deciding whether some $w \in \{a, \dots, z\}^*$ is the imperfect of an english verb.
 - Problem of deciding whether a string $w \in (\{A, \dots, H\} \times \{1, \dots, 8\} \times \{A, \dots, H\} \times \{1, \dots, 8\})^*$ describes
 - a game of chess where 'white' has won,
 - a game of chess that has reached a position where 'white' can enforce a win.

Questions concerning a given problem are in particular

- whether it can be solved by a computer,
- if so, what would be the best algorithm to do so,
- how much computational effort this requires.