St. Pölten University of Applied Sciences

/informatik & security

/fh/// st. pölten

Formal Logics

Theoretical Computer Science

Dipl.-Ing. Hubert Schölnast, BSc
October 04, 2021

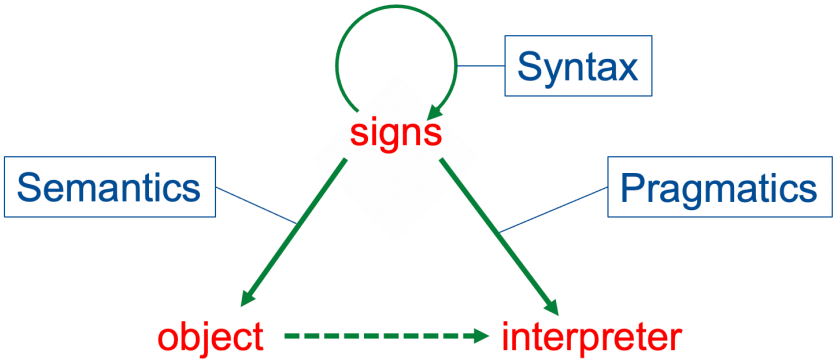
St. Pölten University of Applied Sciences, Campus-Platz 1, 3100 St. Pölten, T: +43 (2742) 313 228, F: +43 (2742) 313 228-339, E: csc@fhstp.ac.at, I: www.fhstp.ac.at

Table of Contents

1 Propositional Logic

1.1 Propositional Formulas

Remember this picture from the lecture about semiotics:



We still will strictly separate syntax (how can signs be arranged) from semantics (what is the meaning of the formula that is made from these signs).

1.1.1 A formal language for propositional formulas

We start in the world of syntax only, i.e. for a moment, we don't care about the meaning.

There is a language of propositional formulas, and each word, that belongs to this language is a propositional formula. We could define this language with a grammar as described in the textbook about formal languges, but this here is a little bit simpler:

Logic variables

Any logic variable is by itself a propositional formula. They are often written as lowercase Latin characters ($a, b, c, \cdots$), but for our purposes it's easier to use always the same character (an a) with indices ($a_1, a_2, a_3, \cdots$).

Transition relation

If ψ and φ are propositional formulas, these sequences of signs are propositional formulas too:

- $(\neg\psi)$ and $(\neg\varphi)$
- $(\psi \vee \varphi)$
- $(\psi \wedge \varphi)$

Convenient signs and transition relations

Some formulas can be written with less signs if you add additional signs and transition rules, for example:

If ψ and φ are propositional formulas, these two sequences can be replaced by each other:

- $(\neg\psi) \vee \varphi$
- $\psi \rightarrow \varphi$

It is possible to add much more of these convenient signs and rules, but none of them are really needed. The 5 signs ¬, ∨, ∧, (and), together with the signs for logic variables and the "original" rules are fully sufficient.

Priority rules

The grammar produces a lot of brackets. If we define priorities, we need less brackets later, when we assign meanings to the words of this language:

- priority 1 (highest priority): (and)
- priority 2: ¬
- priority 3: ∧
- priority 4 (lowest priority): ∨

Note, that in some programming languages ∧ and ∨ have the same priority, and in some others even ∨ has a higher priority than ∧.

1.1.2 Semantics and pragmatics

Let's find a method to assign meaning to the strings. We do this by defining an interpretation function b:

$$b: \{a_i \mid i \in \mathbb{N}\} \longrightarrow \{t, f\}$$

Semantics: Each logical variable is assigned exactly one value from the set $\{t, f\}$

Pragmatics:

- t shall be interpreted as "true"
- f shall be interpreted as "false"

This interpretation is typical for Boolean algebra (introduced by George Boole, 1815-1864, a self-taught English mathematician, logician and philosopher).

Every Boolean interpretation b can be extended to formulas in the following way:

if	then
$b(\psi) = f$	$b(\neg\psi) = t$
$b(\psi) = t$ or $b(\varphi) = t$ (or if the interpretations of both formulas are t)	$b((\psi \vee \varphi)) = t$
$b(\psi) = t$ and $b(\varphi) = t$	$b((\psi \wedge \varphi)) = t$

if	then
$b(\psi) = t$	$b(\neg\psi) = f$
$b(\psi) = f$ and $b(\varphi) = f$	$b((\psi \vee \varphi)) = f$
$b(\psi) = f$ or $b(\varphi) = f$ (or if the interpretations of both formulas are f)	$b((\psi \wedge \varphi)) = f$

1.1.3 Truth tables

These semantic rules can be written clearer in truth tables. (See document "Truth Tables")

1.2 Basic concepts

1.2.1 satisfiable

A formula ψ (a string that is a word of the language of propositional formulas) is called "satisfiable" if $b(\psi) = t$ for **at least one** interpretation of ψ.

In other words:

If you can find at least one combination of interpretations for the logical variables in ψ, such that the interpretation of ψ is t, then ψ is satisfiable.

If ψ is not equivalent to $\perp$, it is satisfiable. Only if ψ is equivalent to $\perp$, it is unsatisfiable.

Examples:

Logical variables are a_1 and a_2.

S1. Is a_1 satisfiable?
This is trivial: Choose the interpretation $b(a_1) = t$ for the variable a_1 and then the interpretation of the whole formula a_1 which is $b(a_1)$ will also be t:

$$b(a_1) = t \quad \Longrightarrow \quad b(a_1) = t$$

So, the answer is: yes!

S2. Is $\neg a_1$ satisfiable?
This is also (almost) trivial: Choose the interpretation $b(a_1) = f$ for the variable a_1 and then the interpretation of the whole formula $\neg a_1$ which is $b(\neg a_1)$ will become t:

$$b(a_1) = f \quad \Longrightarrow \quad b(\neg a_1) = t$$

So, the answer is: yes!

S3. Is $\big((\neg a_1) \wedge a_1\big)$ satisfiable?
The variable a_1 appears twice in this formula. If this is the case for any variable, then all instances of this variable have the same interpretation.

Let's try the interpretation $b(a_1) = t$:

In this case we have $b(\neg a_1) = f$ and therefore also $b\big((\neg a_1)\big) = f$ but we also have $b(a_1) = t$. The last line in the table in 1.1.2 says, that the interpretation of $(\psi \wedge \varphi)$ is f if either $b(\psi) = f$ or $b(\varphi) = f$ (or if the interpretations of both formulas are f). Here we have

$$\psi = (\neg a_1)$$
$$b(\psi) = b\big((\neg a_1)\big) = f$$

and therefor

$$b\big((\psi \wedge \varphi)\big) = b\Big(\big((\neg a_1) \wedge a_1\big)\Big) = f$$

But we have another chance. We can also try $b(a_1) = f$. Then $b\big((\neg a_1)\big) = t$ but $b(a_1) = f$ and so we again have $b\big((\psi \wedge \varphi)\big) = b\Big(\big((\neg a_1) \wedge a_1\big)\Big) = f$

So, the answer is: no!

1.2.2 valid

A formula ψ (a string that is a word of the language of propositional formulas) is called "valid" if $b(\psi) = t$ for **all** interpretations of ψ.

In other words:

If you can find at least one combination of interpretations for the logical variables in ψ, such that the interpretation of ψ is f, then ψ is **not** valid.

If ψ is equivalent to $\top$, it is valid. If ψ is not equivalent to $\top$, it is not valid.

satisfiable vs. valid

To be satisfiable, it is sufficient to find at least one interpretation of the variables where the whole formula is interpreted as t. To be valid, this must be the case for every possible interpretation of the variables. Therefore, all valid formulas are satisfiable. If a formula cannot be satisfied, it cannot be valid.

Examples:

Logical variables are a_1 and a_2.

V1. Is a_1 valid?
If we interpret the variable a_1 as f then the interpretation of the whole formula will also be f:

$$b(a_1) = f \quad \Longrightarrow \quad b(a_1) = f$$

So, we found an interpretation that makes the formula to be f. This means, that not all interpretations of a_1 will result in an interpretation of the formula that is t. So, this formula not valid.
The answer is: no!

V2. Is $\big((\neg a_1) \wedge a_1\big)$ valid?
We already have seen, that this formula is not satisfiable, so can't be valid.
The answer is: no!

V3. Is $\big((\neg a_1) \vee a_1\big)$ valid?
Let's try the interpretation $b(a_1) = t$:

In this case the interpretation of $\neg a_1$ is f and therefor also $b\big((\neg a_1)\big) = f$. The 2nd line in the table in 1.1.2 says, that the interpretation of $(\psi \vee \varphi)$ is t if either $b(\psi) = t$ or $b(\varphi) = t$ (or if both interpretations are t). Here we have

$$\varphi = a_1$$
$$b(\varphi) = b(a_1) = t$$

and therefor

$$b\big((\psi \vee \varphi)\big) = b\Big(\big((\neg a_1) \vee a_1\big)\Big) = t$$

If we try $b(a_1) = f$ then $b\big((\neg a_1)\big) = t$ (note, that $b\big((\neg a_1)\big) = b(\psi)$) and so we again have $b\big((\psi \vee \varphi)\big) = b\Big(\big((\neg a_1) \vee a_1\big)\Big) = t$
We tried all possible interpretations, and in all cases the interpretation of the whole formula was t. And this means, that this formula is valid:
The answer is: yes!

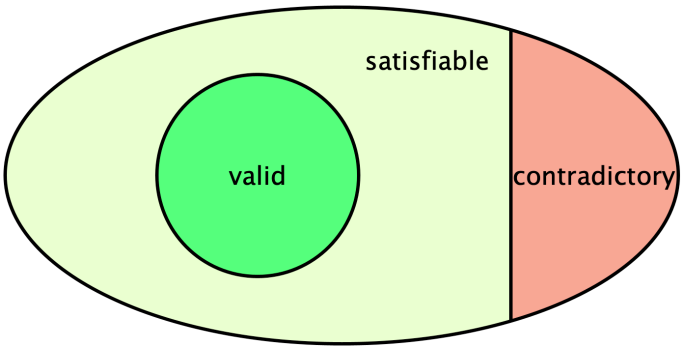
1.2.3 contradictory

A formula ψ (a string that is a word of the language of propositional formulas) is called "contradictory" if $b(\psi) = t$ for **no** interpretation of ψ.

$((\neg a_1) \wedge a_1)$ is a contradictory formula, because both, $b(a_1) = t$ and $b(a_1) = f$, lead to $b\left(((\neg a_1) \wedge a_1)\right) = f$

All other examples shown above are not contradictory.

1.2.4 Summary



1.3 How can we check that?

1.3.1 Truth tables

- A formula is **satisfiable** if (and only if) the truth table contains t in at least 1 row of the formula column.
- A formula is **valid** (a tautology) if (and only if) the truth table contains t in all rows of the formula column.
- A formula is **contradictory** if (and only if) the truth table contains f in all rows of the formula column.

S1, V1

a_1	a_1
t	t
f	f

✅ satisfiable
❌ not valid
❌ not contradictory

S2

a_1	$(\neg a_1)$
t	f
f	t

✅ satisfiable
❌ not valid
❌ not contradictory

S3, V2

a_1	$(\neg a_1) \wedge a_1$
t	t
f	t

✅ satisfiable
✅ valid
❌ not contradictory

V3

a_1	$(\neg a_1) \vee a_1$
t	f
f	f

❌ not satisfiable
❌ not valid
✅ contradictory

New examples, with more than just 1 logic variable

a_1	a_2	$(\neg(a_1 \wedge a_2))$
t	t	f
t	f	t
f	t	t
f	f	t

✅ satisfiable
❌ not valid
❌ not contradictory

a_1	a_2	$\big(((\neg a_1) \vee (a_1 \wedge a_2)) \vee (\neg a_2)\big)$
t	t	t
t	f	t
f	t	t
f	f	t

✅ satisfiable
✅ valid
❌ not contradictory

a_1	a_2	a_3	$\Big(\neg\big(((\neg a_1) \vee (a_1 \wedge a_2)) \vee ((\neg a_2) \vee (\neg a_3))\big)\Big)$
t	t	t	f
t	t	f	f
t	f	t	f
t	f	f	f
f	t	t	f
f	t	f	f
f	f	t	f
f	f	f	f

❌ not satisfiable
❌ not valid
✅ contradictory

formula =

$$\Big(\neg\Big(\big((a_1 \wedge (\neg a_2)) \vee \neg(a_3 \wedge (\neg a_4))\big) \wedge \big(\neg\big((a_5 \wedge (\neg a_6)) \vee \neg(a_7 \wedge (\neg a_8))\big)\big)\Big) \wedge \big((a_9 \wedge (\neg a_{10})) \wedge (a_1 \wedge a_5)\big)\Big)$$

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	formula
t	t	t	t	t	t	t	t	t	t	f
…	…	…	…	…	…	…	…	…	…	f
t	t	t	f	t	t	t	f	t	t	f
t	t	t	f	t	t	t	f	t	f	t
t	t	t	f	t	t	t	f	f	t	f
…	…	…	…	…	…	…	…	…	…	f
f	f	f	f	f	f	f	f	f	f	f

✅ satisfiable
❌ not valid
❌ not contradictory

The truth table for 10 logical variables has 1024 rows. The formula is t in only 1 row and false in all other rows.

A simple hardware chip for a safety-critical system with 3 redundant controllers with 20 Boolean input values per controller contains 60 variables. The truth table of the logical formula that describes this system has $1.15 \cdot 10^{18}$ lines. More complex chips may need thousands of logical variables. It is not possible to check such a huge truth table in reasonable time.

1.3.2 By proof

You work purely syntactically, and you apply some of the production rules of the grammar of propositional logic to parts of the logic formula to make it shorter, so that you can reduce it to known facts.

Direct proof

This shall be explained on an example:

- A: $(a_1 \wedge (a_2 \vee a_3))$
- B: $((a_1 \wedge a_2) \vee (a_1 \wedge a_3))$

We want to proof that, if formula A is true, then also formula B must be true.

We only have 3 logic variables, so the both truth table would have just 8 lines, and it would be easy to check if the columns of both formulas are equal or not, but we want to do it with a direct proof here:

If $b\big((a_1 \wedge (a_2 \vee a_3))\big) = t$ then $b(a_1) = t$ and also $b\big((a_2 \vee a_3)\big) = t$.

If $b\big((a_2 \vee a_3)\big) = t$ then either $b(a_2) = t$ or $b(a_3) = t$ (or both are true).

Case 1: $b(a_2) = t$

We found so far $b(a_1) = t$ and $b(a_2) = t$. So, in B also $(a_1 \wedge a_2)$ must be true:

$b\big((a_1 \wedge a_2)\big) = t$

In this case the interpretation of $(a_1 \wedge a_3)$ doesn't matter. We have:

$b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = t$

Case 2: $b(a_3) = t$

Now we know $b(a_1) = t$ and $b(a_3) = t$. So, $b\big((a_1 \wedge a_3)\big) = t$ and therefor:

$b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = t$

qed.

Indirect proof

We use the same example. In an indirect proof we assume that the conjecture is wrong. If this leads to a contradiction, the opposite must be true, which means it must be true.

So, we assume that there is an interpretation for all logic variables, such that $b\left(\big(a_1 \wedge (a_2 \vee a_3)\big)\right) = t$ and $b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = f$.

We assume $b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = f$. That means, that $b\big((a_1 \wedge a_2)\big) = f$ and also $b\big((a_1 \wedge a_3)\big) = f$. We again have two cases:

Case 1: $b(a_1) = f$

In this case the interpretations of a_2 and a_3 doesn't matter, but if $b(a_1) = f$ then it's impossible to have $b\left(\big(a_1 \wedge (a_2 \vee a_3)\big)\right) = t$. So, case 1 is impossible.

Case 1: $b(a_1) = t$

If $b(a_1) = t$ and $b\big((a_1 \wedge a_2)\big) = f$ then $b(a_2) = f$

If $b(a_1) = t$ and $b\big((a_1 \wedge a_3)\big) = f$ then $b(a_3) = f$

If $b(a_2) = f$ and $b(a_3) = f$ then $b\big((a_2 \vee a_3)\big) = f$ but then also $b\left(\big(a_1 \wedge (a_2 \vee a_3)\big)\right) = f$

So, also case 2 is impossible.

All cases are impossible, so the assumption $b\left(\big(a_1 \wedge (a_2 \vee a_3)\big)\right) = t$ and $b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = f$ must be wrong, and it must be true, that $b\left(\big((a_1 \wedge a_2) \vee (a_1 \wedge a_3)\big)\right) = t$ if $\left(\big(a_1 \wedge (a_2 \vee a_3)\big)\right) = t$.

qed.

There are more proof methods, not shown here.

1.4 Propositional logic is decidable

Every propositional logic formula has a well-defined interpretation if the interpretations of the variables are given. So it is always possible to decide unambiguously if a propositional logic formula is satisfiable, valid or contradictory.

This can be done by constructing truth tables or with special proof systems for propositional logic formulas called "SAT solvers".

2 Predicate logic

In natural language there is more than „not“, „and“, „or“ and „if – then“. We want to be able to express not only elementary propositions, but propositions about quantities (sets), too. This can be done in predicate logic.

2.1 Predicates

A predicate is a Boolean function, i.e. a function with the codomain $\{t, f\}$.

2.1.1 Examples

Let x be a variable. Then we can define the unary predicate

- $St(x)$

and we can define for this particular predicate, that the value of this predicate is t if x is a student. If x is not a student (if x is a person that is not a student, or if x is an apple, the planet Mars, the number 6 or anything else), then the value of this predicate is f. So, the value of each predicate is always predefined. It is t if the variable is an element of a given set (here the set of all students), otherwise it's f.

We can define another unary predicate:

- $T(x)$ is t if x is a teacher (and it's f if x is something else but a teacher)

Let's try a binary predicate:

- $O(x, y)$ is t if both, x and y are persons and if x is older than y.

2.2 Quantifiers

There are two quantifiers:

- ∀ the all-quantifier
- ∃ the existence-quantifier

2.2.1 Examples

$$\forall x \Big(St(x) \rightarrow \big(\exists y \big(T(y) \wedge O(y,x) \big) \big) \Big)$$

Meaning: For all x holds, that if x is a Student, then there exists a y, such that y is a Teacher and y (teacher) is Older than x (student).

Or shorter: Every student has at least 1 teacher who is older than the student.

This is a predicate logic formula, and this formula again can be interpreted:

$$b\Big(\forall x \Big(St(x) \rightarrow \big(\exists y \big(T(y) \wedge O(y,x) \big) \big) \Big) \Big) \in \{t, f\}$$

But before we talk about interpretation, let's introduce ...

2.3 Predicate logic functions

Let's say, we have these predicates

- $Ch(x)$ is t if x is a child
- $M(x,y)$ is t if x and y are persons and if x is the mother of y.
- $O(x,y)$ is t if x and y are persons and if x is older than y.

Then we can write this predicate logic formula

$$\forall a \Big(\forall b \Big(\big(Ch(a) \wedge M(b,a) \big) \rightarrow O(b,a) \Big) \Big)$$

(Meaning: Every mother is older than all of her children.)

This can be simplified by the use of a predicate logic function:

$$\forall a \big(Ch(a) \rightarrow O(\mathrm{Mother}(a), a) \big)$$

The codomain of the function $\mathrm{Mother}(x)$ is not $\{t, f\}$, so this function is not a predicate! The codomain of this particular function is the set of all mothers (or any superset of this set). For every x that can be fed into this function, it will return this persons mother. This of course is only possible if x is a parson, so the domain of $\mathrm{Mother}(x)$ is not the class of everything, but only the set of persons.

2.4 Operators

We have the same operators as in prepositional logic:

- ¬ not
- ∧ and
- ∨ (inclusive) or
- → if ... then (this is a convenient operator. It can be written as a combination of ¬, ∧ and ∨, but then the formulas would become much more complicated und hard to understand)

2.5 Syntax

The language of predicate logic consists of Terms und Formulas only. The alphabet of this language contains the following sets:

- P the set of predicates
- F the set of functions
- C the set of constants (= nullary functions)
- V the set of variables

2.5.1 Term

A term is one of these things:

- $x \in V$ a variable
- $c \in C$ a constant
- $f(t_1, t_2, \cdots, t_n) \in F$ a function where $t_1, t_2, \cdots, t_n$ are terms.

Example of a term

We define

$$V = \{x, y\}$$
$$C = \mathbb{N}$$
$$F = \{+, -, \cdot, \mathrm{sqr}\}$$

Then this is a correct term:

$$\cdot\big(-\big(2, +(\mathrm{sqr}(x), y)\big), x\big)$$

We can interpret this term as

$$\big(2 - (x^2 + y)\big) \cdot x$$

2.5.2 Formula

A formula is one of these things:

- $p(t_1, t_2, \cdots, t_n) \in P$ a predicate where $t_1, t_2, \cdots, t_n$ are terms.
- $(\neg\varphi)$ where φ is a formula.
- $(\varphi \wedge \psi)$ where φ and ψ are formulas.
- $(\varphi \vee \psi)$ where φ and ψ are formulas.
- $(\varphi \rightarrow \psi)$ where φ and ψ are formulas.
- $\big(\forall x(\varphi)\big)$ where x is a variable ($x \in V$) and φ is a formula.
- $\big(\exists x(\varphi)\big)$ where x is a variable ($x \in V$) and φ is a formula.

We again can define priorities when we interpret predicate logic functions:

- Priority 1 (highest priority): (and)
- Priority 2: ¬, ∀ and ∃
- Priority 3: ∧ and ∨
- Priority 4: →

Example of a formula

we define:

- Walter is a constant
- $f(a)$ is a unary function, yielding the father of a
- $S(a,b)$ is a binary predicate for "a is son of b"
- $B(a,b)$ is a binary predicate for "a is brother of b"

Then this is a formula:

$$\Big(\forall x \Big(S\big(x, f(Walter)\big) \rightarrow B(x, Walter)\Big)\Big)$$

Attention! The interpretation of this formula is only t if x is not Walter!

2.5.3 bound variables

We must make a difference between free variables and bound variables. For example in the formula:

$$\forall x\big(B(x,y) \rightarrow male(x)\big)$$

x is a bound variable, it is bound by $\forall x$. y on the other hand is a free (unbounded) variable.

Only free variables can be substituted by terms. And in this case only terms are allowed, that don't contain bound variables!

Given a formula φ, a term t and a variable x:

If we substitute the variable x in the formula φ by the term t, we write: $\varphi[t/x]$

This is allowed only if x is a free variable!!!

2.6 Examples

Let's define some predicates:

- $\text{prime}(x)$ x is a prime number
- $\text{book}(x)$ x is a book
- $\text{person}(x)$ x is a person
- $\text{reads}(x,y)$ x reads y
- $\leq (x,y)$ $x \leq y$
- $= (x,y)$ x and y are identic

Then we can write some predicate logic formulas:

1. There is a smallest prime number.

$\exists n \Big(\text{prime}(n) \wedge \forall m\big(\text{prime}(m) \rightarrow \leq (n,m)\big)\Big)$

2. Every book is read by at least one human.

$\forall b \Big(\text{book}(b) \rightarrow \exists m\big(\text{person}(m) \wedge \text{reads}(m,b)\big)\Big)$

3. Every human reads at least two books

$$\forall m \left(person(m) \rightarrow \exists b_1 \exists b_2 \left(book(b_1) \wedge book(b_2) \wedge \neg = (b_1, b_2) \wedge reads(m, b_1) \wedge reads(m, b_2)\right)\right)$$

2.7 satisfiable – valid – contradictory

These terms are defined equivalent to those in propositional logic. But in predicate logic truth tables cannot be constructed – not even in theory – because we would have to test all possible values of all variables of a formula:

$\forall x$ (...) we must test for all x (infintely many!)

To test, if a predicate logic formula is satisfiable, valid or contractionary, there is only one way: A proof.

2.7.1 Proof rules

These rules are just a selection. There are much more rules

- Equality

  If two terms t_1 and t_2 are equal we can substitute a free variable x in a formula φ by t_1 or t_2 as we like and it does not change anything in the formula.

  $$t_1 = t_2 \quad \rightarrow \quad \varphi[t_1/x] = \varphi[t_2/x]$$

- Elimination of ∀

  If a formula is valid for all x then x may be substituted by any arbitrary term.

  $$\forall x(\varphi) \quad \rightarrow \quad \varphi[t/x]$$

- Introduction of ∃

  If a formula is valid for any term t then there must be at least one x, for which the formula is valid.

  $$\varphi[t/x] \quad \rightarrow \quad \exists x(\varphi)$$

There are also proof rules for the introduction of ∀ and the elimination of ∃ but they are a little bit more complicated.

The next rules all seem quite logic if you think about them

- $\neg \forall x(\varphi) \Leftrightarrow \exists x(\neg \varphi)$
- $\neg \exists x(\varphi) \Leftrightarrow \forall x(\neg \varphi)$
- $\left(\forall x(\varphi)\right) \wedge \left(\forall x(\psi)\right) \Leftrightarrow \left(\forall x(\varphi \wedge \psi)\right)$
- $\left(\exists x(\varphi)\right) \vee \left(\exists x(\psi)\right) \Leftrightarrow \left(\exists x(\varphi \vee \psi)\right)$
- $\exists x\left(\exists y(\varphi)\right) \Leftrightarrow \exists y\left(\exists x(\varphi)\right)$
- $\forall x\left(\forall y(\varphi)\right) \Leftrightarrow \forall y\left(\forall x(\varphi)\right)$

The next rules apply only if x is not free in ψ:

- $\forall x(\varphi) \wedge \psi \Leftrightarrow \forall x(\varphi \wedge \psi)$
- $\forall x(\varphi) \vee \psi \Leftrightarrow \forall x(\varphi \vee \psi)$
- $\exists x(\varphi) \wedge \psi \Leftrightarrow \exists x(\varphi \wedge \psi)$
- $\exists x(\varphi) \vee \psi \Leftrightarrow \exists x(\varphi \vee \psi)$

2.8 Universe of discourse

In logic and philosophy of language, a universe of discourse is understood as the totality of objects to which statements refer. Such statements are meaningful only if the meaning of "object" is restricted to a particular domain, the universe of discourse. The extent and nature of the restriction depend on the content and context of the statements. Therefore, there is not only one universe of discourse, but different universes of discourse.

The term Universe of Discourse goes back to Augustus De Morgan (1847) and refers to the range of objects (in the broadest sense) that are to be talked about at all.

Misunderstandings and disputes often arise in logic, as in everyday life, because people talk at cross purposes about different things. Someone claims, for example, that there are no winged horses. His counterpart rejects this with the reference to the Pegasus. Both move mentally in different worlds. Their dispute can be settled if they agree on a common universe of discourse, i.e. negotiate what the talk (discourse) should be about, whether only physically existing horses or also mythical creatures.

So, in predicate logic we need a Universe of Discourse $\mathcal{D}$ (what we are talking about) and a function that maps the (syntactic) elements of the predicate logic onto the Universe of Discourse $\mathcal{D}$.

- Every constant maps to an element of $\mathcal{D}$.
- Every n-ary function corresponds to a concrete n-ary function in $\mathcal{D}$.
- Every n-ary predicate corresponds to a set of n-tuples over the elements of $\mathcal{D}$ (those tuples, for which the predicate is valid).

The Universe of Discourse together with this mapping is called a **model**.

2.8.1 Example

Assume a predicate logic contains a unary function $m(x)$, and a binary predicate $j(x, y)$.

One model (interpretation, semantics) could be:

$$\mathcal{D} = \{\text{Max}, \text{Bob}, \text{Eva}, \text{Amy}\}$$

$$m(x) = \{(\text{Max}, \text{Eva}), (\text{Bob}, \text{Eva}), (\text{Amy}, \text{Eva}), (\text{Lea}, \text{Amy})\}$$

$$j(x, y) = \{(\text{Max}, \text{Eva}), (\text{Bob}, \text{Eva}), (\text{Amy}, \text{Eva}), (\text{Max}, \text{Bob}), (\text{Bob}, \text{Lea})\}$$

In this model $\forall x\big(m(x)\big) \rightarrow j\big(x, m(x)\big)$ is valid.

x	$m(x)$	$j\big(x, m(x)\big)$
Max	Eva	✅
Bob	Eva	✅
Eva	*(not defined)*	*(doesn't matter)*
Amy	Eva	✅

If we use a different universe of discourse, we get this situation:

- $\mathcal{D} = \{\text{Max}, \text{Bob}, \text{Eva}, \text{Amy}, \text{Lea}\}$

x	$m(x)$	$j\big(x, m(x)\big)$
Max	Eva	✅
Bob	Eva	✅
Eva	*(not defined)*	*(doesn't matter)*
Amy	Eva	✅
Lea	Amy	❌

2.9 Decidability

Remember, that propositional logic was decidable. But predicate logic is not. It is semidecidable. This means: If a predicate logic formula is valid, it is always possible to proof that this is the case. But if a predicate logic formula is invalid (if it's satisfiable but not valid or even contradictory), then this can be proven only for some of these invalid formulas, but not for all.

There is an algorithm that allows to enumerate all valid predicate logic formulas. And this means that the cardinality of this set is $\aleph_0$ which is the cardinality of all Turing machines. So, there is a Turing machine for each valid predicate logic formula.

But it has been proven, that the cardinality of all invalid predicate logic formulas is greater than $\aleph_0$. So, there are more invalid predicate logic formulas than Turing machines, and this means, that there must exist invalid predicate logic formulas for which there is no Turing machine that could halt and tell that it's invalid.