Formal Method in Software Engineering

Michael Winter

December 11, 2014

Contents

Chapter 1

Mathematical Preliminaries

1.1 Induction

A very important proof method in mathematics (and computer science) is the principle of induction. The most commonly known form of induction is mathematical induction. In this section we want to investigate a method called well-founded induction. This method is very powerful and implies most principles of induction.

Definition 1.1.1 *A well-founded relation is a binary relation $\prec$ on a set A such that there are no infinite descending chains $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$. If $a \prec b$ we say that a is a predecessor of b.*

A well-founded relation is necessarily irreflexive, i.e. there is no element $a \in A$ with $a \prec a$, since, otherwise, $\cdots \prec a \prec \cdots \prec a \prec a$ is an infinite chain.

We denote the reflexive closure of $\prec$ by $\preceq$, i.e.

$$a \preceq b :\Longleftrightarrow\ a = b \text{ or } a \prec b.$$

Notice that we do not require that $\prec$ is transitive. However, well-founded relations may be characterized in terms of minimal elements.

Theorem 1.1.2 *Let $\prec$ be a binary relation on a set A. Then $\prec$ is well-founded iff*[1] *any non-empty subset $B \subseteq A$ has a minimal element, i.e. an element $b \in B$ with*

$$\forall c : (c \prec b \Rightarrow c \notin B).$$

[1] We write iff as an abbreviation for if and only if.

Proof. $\Leftarrow$: Suppose every non-empty subset of A has a minimal element, and assume $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$ is an infinite chain. Then the set $B = \{a_i \mid i \geq 0\}$ does not have a minimal element, a contradiction.

$\Rightarrow$: Assume B is a non-empty subset of A. Construct a chain as follows: Let a_0 be an arbitrary element of B. Inductively, assume a chain $a_n \prec \cdots \prec a_0$ of n elements has been constructed. If there is a $b \in B$ with $b \prec a_n$ take $a_{n+1} = b$. If not stop the construction. The constructed chain must be finite since $\prec$ is well-founded, i.e. is of the form $a_n \prec \cdots \prec a_0$. a_n is the required minimal element since no element of B is a predecessor of a_n by construction. □

Now, we are going to formulate (and prove) the principle of well-founded induction.

Theorem 1.1.3 (Principle of well-founded induction) *Let $\prec$ be a well-founded relation on a set A, and P be a property of elements of A. Then we have*

$$\forall a \in A : P(a) \text{ iff } \forall a \in A : ([\forall b \prec a : P(b)] \Rightarrow P(a)).$$

Proof. $\Rightarrow$: This is trivial since P holds for all elements of A.

$\Leftarrow$: Suppose $\forall a \in A : ([\forall b \prec a : P(b)] \Rightarrow P(a))$, and assume that P does not hold for an element $a \in A$. Then the set $\{a \in A \mid \neg P(a)\}$ is not empty, and has, by the previous theorem, a minimal element m. Now, let $b \in A$ be an arbitrary element with $b \prec m$. Since m is minimal b must have property P, implying that all predecessors of m satisfy P. We conclude $P(m)$, a contradiction. □

Notice that the base case is given by those elements that do not have a predecessor.

If we take $n \prec m$ iff $n + 1 = m$, i.e. the successor relation on natural numbers, the principle of well-founded induction specializes to mathematical induction.

Later we will use structural induction on expressions and derivations. Both principles are again special cases of well-founded induction.

Chapter 2

First-Order Logic

In this chapter we want to study first-order logic as a formal system, i.e., as a system with specific rules that can be implemented on a computer. In first-order logic one is allowed to combine basic statements using logical connectives and to quantify over entities.

2.1 Syntax

In order to provide a suitable language to talk about elements of the domain of interest we require the following components:

1. X a set of variables.
2. F a set of function symbols. Each symbol has its arity.
3. P a set of predicate symbols. Each symbol has its arity.

0-ary functions symbols are called constant symbols. Such a symbol corresponds to a function requiring no parameter at all, i.e., the function can be identified with the element it returns. Similar the propositional variable of propositional logic can be identified with 0-ary predicate symbols so that first-order logic becomes a extension of propositional logic.

Definition 2.1.1 *The set* Term *of terms is recursively defined by the following.*

1. *Each variable $x \in X$ is a term, i.e., $X \subseteq$ Term.*

2. *If $f \in F$ is an n-ary function symbol and $t_1, \ldots, t_n \in$ Term are terms, then $f(t_1, \ldots, t_n) \in$ Term.*

Examples of terms are $f(x, y, z)$ or $f(f(x, x, x), f(y, y, y), z)$ assuming that f is a ternary function symbol and x, y, z are variables.

Definition 2.1.2 *The set* FOL *of first-order formulas (or formulas) is recursively defined by the following.*

1. *If $p \in P$ is an n-ary predicate symbol and $t_1, \ldots, t_n \in$ Term are terms, then $p(t_1, \ldots, t_n) \in$ FOL. Formulas of this kind are called atomic formulas.*
2. *$\bot$ is a formula, i.e, $\bot \in$ FOL.*
3. *If $\varphi \in$ FOL then $\neg\varphi \in$ FOL.*
4. *If $\varphi_1, \varphi_2 \in$ FOL then*
   (a) *$\varphi_1 \wedge \varphi_2 \in$ FOL and*
   (b) *$\varphi_1 \vee \varphi_2 \in$ FOL and*
   (c) *$\varphi_1 \to \varphi_2 \in$ FOL.*
5. *If $\varphi \in$ FOL and $x \in X$ then*
   (a) *$\forall x{:}\varphi \in$ FOL and*
   (b) *$\exists x{:}\varphi \in$ FOL.*

The previous definition defines the set of formulas by giving a set of rules which may be applied to a base set finitely many times. We also say that the set FOL is defined recursively by those rules. A set defined in such a way always provides a principle of induction (see also Chapter 1). In the example of FOL that principle reads as follows. If we want to show that a certain property N is true for all elements in FOL it is sufficient to

Base case: show the property N for atomic formulas $p(t_1, \ldots, t_n)$ and the special formula $\bot$;

Induction step I: show the property N for $\neg\varphi$ by assuming that it is already true for φ (induction hypothesis);

Induction step II: show the property N for $\varphi_1 \otimes \varphi_2$ for $\otimes \in \{\wedge, \vee, \rightarrow\}$ by assuming that it is already true for φ_1 and φ_2 (induction hypothesis).

Induction step III: show the property N for $Qx : \varphi$ for $Q \in \{\forall, \exists\}$ by assuming that it is already true for φ (induction hypothesis).

The base case, in particular in the case of an atomic formula $p(t_1, \ldots, t_n)$, may require another induction based on the structure of terms. This time the base case is the case where the term is a variable, and the induction step covers the case that the term is a function symbol applied to n terms.

We adopt certain precedence rules of the logical symbols. $\neg$, $\forall$, and $\exists$ bind more tightly than $\wedge$, $\wedge$ tighter than $\vee$, and $\vee$ tighter than $\rightarrow$. For example, the proposition

$$p \wedge q \vee \neg r \rightarrow p$$

has to be read as

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

Last but not least, we will use the following abbreviations. We write $\top$ for $\neg\bot$ and $\varphi_1 \leftrightarrow \varphi_2$ for $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$.

In the case of binary function and predicate symbols we will also use infix notation. For example, instead of writing $+(x, y)$ and $\leq (x, y)$ we use $x + y$ and $x \leq y$.

Example 2.1.3 *In this example we want to express some properties of the natural numbers in first-order logic. For this purpose we assume that* 1 *is a constant symbol (*0*-ary function symbol),* s *a unary function symbol, and* $=$ *a binary predicate symbol. With the interpretation in mind that* 1 *denotes the number one,* s *is the successor function, and* $=$ *denotes equality we may state the following formulas:*

$$\forall x{:}\forall y{:}(s(x) = s(y) \rightarrow x = y)$$
$$\forall x{:}(\neg(x = 1) \rightarrow \exists y{:}x = s(y)).$$

Notice that we cannot express the principle of induction since it quantifies over all properties. Such a statement is covered by second-order logic.

We adapt the usual conventions for some negated atomic formulas. For example, instead of writing $\neg(x = y)$ and $\neg(x \leq y)$ we use $x \neq y$ and $x \not\leq y$. In addition, we will group quantifications, i.e., we write $\forall x, y, z{:}\varphi$ instead of $\forall x{:}\forall y{:}\forall z{:}\varphi$.

Definition 2.1.4 *An occurrence of a variable $x \in X$ in a formula is called bounded iff it is in a subformula of the form $\forall x{:}\varphi$ or $\exists x{:}\varphi$. An occurrence is called free iff it is not bounded.*

Consider the formula

$$\forall x{:}P(x) \wedge \exists y{:}Q(x,y).$$

The occurrence of x in $P(x)$ and the occurrence of y are bounded. The second occurrence of x in $Q(x,y)$ is free. Another example is

$$\forall y{:}(y|x \wedge y \neq 1 \rightarrow x = y).$$

Here all occurrences of y are bounded, and all occurrences of x are free. With the standard interpretation the formula above expresses that x is prime.

Since variables are place holders we need some means to replace them with a concrete object. For example, we may want to replace x in the last formula by the constant symbol 2 in order to state that 2 is prime. In general we want to replace a variable by a term. Unfortunately, we have to be careful because of some undesired side effects of that operation. If we replace x by y in the last example we get

$$\forall y{:}(y|y \wedge y \neq 1 \rightarrow y = y).$$

This formula does not stand for 'y is prime'. The problem is that the term we going to substitute contains a variable y, and that a free occurrence of x is under the scope of $\forall y{:}$ or $\exists y{:}$. The variable is free so that any variable contained in the term we are going to substitute should also be free.

Definition 2.1.5 *Let $x \in X$ be a variable, and $\varphi \in$ FOL be a formula. A $t \in$ Term is called free for x in φ iff no free occurrence of x is in a subformula $\forall y{:}\varphi'$ or $\exists y{:}\varphi'$ for a variable y occurring in t.*

Now we are ready to introduce the notion of substitution.

Definition 2.1.6 *Let $x \in X$ be a variable, $t, t' \in$ Term be terms, and $\varphi \in$ FOL be a formula.*

1. *By $t'[t/x]$ we denote the result of replacing all occurences of x in t' by t.*
2. *If t is free for x in φ, then we denote by $\varphi[t/x]$ the result of replacing any free occurrence of x in φ by t.*

If we write $\varphi[t/x]$ we always assume that t is free for x. Later we will see that this can always be achieved by renaming bounded variables.

2.2 Semantics

Since we are now able to talk about individuals or elements the simple universe of truth values is not rich enough to provide a suitable interpretation of the entities of the language.

Definition 2.2.1 *Let F be a set of function symbols, and P be a set of predicate symbols. A model $\mathcal{M}$ consists of the following data:*

1. *$|\mathcal{M}|$ a non-empty set, called the universe.*
2. *For each function symbol $f \in F$ with arity n a n-ary function $f^{\mathcal{M}} : |\mathcal{M}|^n \to |\mathcal{M}|$.*
3. *For each predicate symbol $p \in P$ with arity n a subset $p^{\mathcal{M}} \subseteq |\mathcal{M}|^n$.*

Notice that constant symbols are interpreted by elements.

Definition 2.2.2 *Let $\mathcal{M}$ be a model. An environment $\sigma : X \to |\mathcal{M}|$ is a function from the set of variables X to the universe of the model. For an environment σ, a variable x, and a value $a \in |\mathcal{M}|$ denote by $\sigma[a/x]$ the environment defined by*

$$\sigma[a/x](y) = \begin{cases} a & \text{iff } x = y, \\ \sigma(y) & \text{iff } x \neq y. \end{cases}$$

We start with the interpretation of a term. Naturally, a term should denote an element so that the interpretation of a term is an element of the universe.

Definition 2.2.3 *Let $\mathcal{M}$ be a model, and σ be an environment. The extension $\bar{\sigma} : \mathrm{Term} \to |\mathcal{M}|$ of σ is defined by:*

1. *$\bar{\sigma}(x) = \sigma(x)$ for every $x \in X$.*
2. *$\bar{\sigma}(f(t_1, \ldots, t_n)) = f^{\mathcal{M}}(\bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_n))$.*

The next step is to define the validity of formulas.

Definition 2.2.4 *Let $\mathcal{M}$ be a model, and σ be an environment. The satisfaction relation $\models_{\mathcal{M}} \varphi[\sigma]$ is recursively defined by:*

1. $\models_{\mathcal{M}} p(t_1, \ldots, t_n)[\sigma]$ *iff* $(\bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_n)) \in p^{\mathcal{M}}$.
2. $\not\models_{\mathcal{M}} \bot[\sigma]$, *i.e., not* $\models_{\mathcal{M}} \bot[\sigma]$.
3. $\models_{\mathcal{M}} \neg\varphi[\sigma]$ *iff* $\not\models_{\mathcal{M}} \varphi[\sigma]$.
4. $\models_{\mathcal{M}} \varphi_1 \wedge \varphi_2[\sigma]$ *iff* $\models_{\mathcal{M}} \varphi_1[\sigma]$ *and* $\models_{\mathcal{M}} \varphi_2[\sigma]$.
5. $\models_{\mathcal{M}} \varphi_1 \vee \varphi_2[\sigma]$ *iff* $\models_{\mathcal{M}} \varphi_1[\sigma]$ *or* $\models_{\mathcal{M}} \varphi_2[\sigma]$.
6. $\models_{\mathcal{M}} \varphi_1 \to \varphi_2[\sigma]$ *iff* $\models_{\mathcal{M}} \varphi_2[\sigma]$ *whenever* $\models_{\mathcal{M}} \varphi_1[\sigma]$.
7. $\models_{\mathcal{M}} \forall x{:}\varphi[\sigma]$ *iff* $\models_{\mathcal{M}} \varphi[\sigma[a/x]]$ *for all* $a \in |\mathcal{M}|$.
8. $\models_{\mathcal{M}} \exists x{:}\varphi[\sigma]$ *iff* $\models_{\mathcal{M}} \varphi[\sigma[a/x]]$ *for some* $a \in |\mathcal{M}|$.

From the definition above we derive some additional notions.

Definition 2.2.5 *Let Σ be a set of formulas, and φ be a formula.*

1. *φ is called valid in the model $\mathcal{M}$ ($\models_{\mathcal{M}} \varphi$) iff $\models_{\mathcal{M}} \varphi[\sigma]$ for all environments σ.*
2. *φ is called valid ($\models \varphi$) iff $\models_{\mathcal{M}} \varphi$ for all models $\mathcal{M}$.*
3. *φ is called satisfiable iff there is a model $\mathcal{M}$ and an environment so that $\models_{\mathcal{M}} \varphi[\sigma]$.*
4. *φ follows from Σ in $\mathcal{M}$ ($\Sigma \models_{\mathcal{M}} \varphi$) iff for all environments σ, whenever $\models_{\mathcal{M}} \psi[\sigma]$ for all $\psi \in \Sigma$, then $\models_{\mathcal{M}} \varphi[\sigma]$.*
5. *φ follows from Σ ($\Sigma \models \varphi$) iff $\Sigma \models_{\mathcal{M}} \varphi$ for all models $\mathcal{M}$.*

Let us consider an example.

Example 2.2.6 *Consider the language and the formulas of Example 2.1.3 again. The first model is the set of natural number $\mathbb{N}$ and the obvious interpretation of the function, constant and predicate symbols, e.g., $s^{\mathbb{N}}(x) = x + 1$. In this case both formulas are valid. The first formula $\forall x{:}\forall y{:}(s(x) = s(y) \to x = y)$ simply says that successor is injective, and the second formula $\forall x{:}(\neg(x = 1) \to \exists y{:}x = s(y))$ says that every element except 1 has a predecessor.*

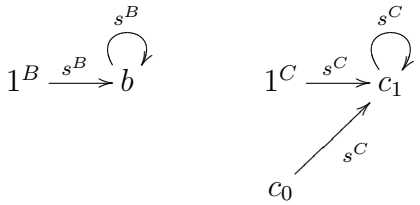
Now, we want to consider several different models. In all cases we will interpret the symbol $=$ by equality. First, consider the model A with universe $\{1^A\}$. The function

symbol s is interpreted by the identity function. This model can be visualized by the following graph:

The identity function is injective and all elements have a predecessor so that both formulas are again satisfied.

For the second model consider again the natural numbers. This time we interpret s by the function $n \mapsto 2n$. This function is injective but all odd numbers are not in the image of that function.

Last but not least, consider the models B and C given by the graphs



Here s^B is not injective but every elements except 1 has a predecessor. In the model C both formulas are not true.

The following lemma states that a formula only depends on the variables that occur free.

Lemma 2.2.7 *Let $t \in \mathrm{Term}$ be a term, $\varphi \in \mathrm{FOL}$ be a formula, and $\mathcal{M}$ be a model.*

1. *If the environments σ_1 and σ_2 coincide on all variables of t, then $\bar{\sigma}_1(t) = \bar{\sigma}_2(t)$.*
2. *If the environments σ_1 and σ_2 coincide on all free variables of φ, then $\models_{\mathcal{M}} \varphi[\sigma_1]$ iff $\models_{\mathcal{M}} \varphi[\sigma_2]$.*

Proof. Both proofs are done by induction.

1. If $t = x$ is a variable we get

$$\bar{\sigma}_1(t) = \sigma_1(x) = \sigma_2(x) = \bar{\sigma}_2(t).$$

If t is of the form $f(t_1, \ldots, t_n)$ with a n-ary function symbol f and terms $t_1, \ldots, t_n$ we conclude

$$\begin{aligned} \bar{\sigma}_1(t) &= f^{\mathcal{M}}(\bar{\sigma}_1(t_1), \ldots, \bar{\sigma}_1(t_n)) \\ &= f^{\mathcal{M}}(\bar{\sigma}_2(t_1), \ldots, \bar{\sigma}_2(t_n)) \qquad \text{by the induction hypothesis} \\ &= \bar{\sigma}_2(t). \end{aligned}$$

2. If $\varphi = p(t_1, \ldots, t_n)$ is an atomic formula then every variable in each term is free in φ so that we conclude $\bar{\sigma}_1(t_i) = \bar{\sigma}_2(t_i)$ for $i \in \{1, \ldots, n\}$ using 1., which immediately implies the assertion.

   If $\varphi = \bot$, then we have $\not\models_{\mathcal{M}} \varphi[\sigma_1]$ and $\not\models_{\mathcal{M}} \varphi[\sigma_2]$ by definition.

   Assume $\varphi = \varphi_1 \wedge \varphi_2$. Then

$$\begin{aligned} \models_{\mathcal{M}} \varphi[\sigma_1] &\Leftrightarrow \models_{\mathcal{M}} \varphi_1[\sigma_1] \text{ and } \models_{\mathcal{M}} \varphi_2[\sigma_1] \\ &\Leftrightarrow \models_{\mathcal{M}} \varphi_1[\sigma_2] \text{ and } \models_{\mathcal{M}} \varphi_2[\sigma_2] && \text{Ind. Hyp.} \\ &\Leftrightarrow \models_{\mathcal{M}} \varphi[\sigma_1]. \end{aligned}$$

   The cases where φ is one of the formulas $\neg\varphi'$, $\varphi_1 \vee \varphi_2$, or $\varphi_1 \to \varphi_2$ are similar to the previous case.

   Assume $\varphi = Qx{:}\varphi'$ with $Q \in \{\forall, \exists\}$. The free variables of φ' are the free variables of φ and the variable x. Consequently, the environments $\sigma_1[a/x]$ and $\sigma_2[a/x]$ for an arbitrary $a \in |\mathcal{M}|$ coincide on all free variables in φ'. We conclude

$$\begin{aligned} \models_{\mathcal{M}} \varphi[\sigma_1] &\Leftrightarrow \models_{\mathcal{M}} \varphi'[\sigma_1[a/x]] && \text{for all/some } a \in |\mathcal{M}| \\ &\Leftrightarrow \models_{\mathcal{M}} \varphi'[\sigma_2[a/x]] && \text{for all/some } a \in |\mathcal{M}| \\ &\Leftrightarrow \models_{\mathcal{M}} \varphi[\sigma_2], \end{aligned}$$

   where the second equivalence is an application of the induction hypothesis. □

The next lemma relates the two notion of substitution and updating an environment. First we want to illustrate it by an example.

Example 2.2.8 *Consider again the formula $\forall y{:}(y|x \wedge y \neq 1 \to x = y)$ using in the standard model $\mathbb{N}$ of the natural numbers, i.e., this formula states that x is prime. Now, consider the term $2 + 3$. On the one hand we could substitute $2 + 3$ for x in the formula, giving $\forall y{:}(y|(2 + 3) \wedge y \neq 1 \to 2 + 3 = y)$, and check its validity for a given environment σ. The formula is true for any environment since it does not contain any free variable. On the other hand, we could first compute the value $\bar{\sigma}(2+3) = \mathbf{5}$ (in order to distinguish terms and natural numbers we use bold symbols for numbers). This time we compute the validity of the original formula with the environment $\sigma[\mathbf{5}/x]$. Once again this result in true.*

Lemma 2.2.9 *Let $x \in X$ be a variable, $t, t' \in$ Term be terms, $\varphi \in$ FOL be a formula, and $\mathcal{M}$ be a model.*

1. $\bar{\sigma}(t'[t/x]) = \overline{\sigma[\bar{\sigma}(t)/x]}(t')$.
2. $\models_{\mathcal{M}} \varphi[t/x][\sigma]$ *iff* $\models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]]$.

Proof. Both assertions are shown by induction.

1. If $t' = y$ we distinguish two cases. If $x = y$ we get

$$\bar{\sigma}(t'[t/x]) = \bar{\sigma}(t) = \overline{\sigma[\bar{\sigma}(t)/x]}(x) = \overline{\sigma[\bar{\sigma}(t)/x]}(t').$$

If $x \neq y$ the environments σ and $\overline{\sigma[\bar{\sigma}(t)/x]}$ coincide on all variables in t'. We use Lemma 2.2.7(1) and conclude

$$\bar{\sigma}(t'[t/x]) = \bar{\sigma}(y) = \overline{\sigma[\bar{\sigma}(t)/x]}(y) = \overline{\sigma[\bar{\sigma}(t)/x]}(t').$$

If $t' = f(t_1, \ldots, t_n)$ we immediately get

$$\begin{array}{lll} & \bar{\sigma}(t'[t/x]) & \\ & = f^{\mathcal{M}}(\bar{\sigma}(t_1[t/x]), \ldots, \bar{\sigma}(t_n[t/x])) & \text{substitution} \\ & = f^{\mathcal{M}}(\overline{\sigma[\bar{\sigma}(t)/x]}(t_1), \ldots, \overline{\sigma[\bar{\sigma}(t)/x]}(t_n)) & \text{induction hypothesis} \\ & = \overline{\sigma[\bar{\sigma}(t)/x]}(t') & \end{array}$$

2. If $\varphi = p(t_1, \ldots, t_n)$ we conclude

$$\begin{array}{lll} & \models_{\mathcal{M}} \varphi[t/x][\sigma] & \\ & \Leftrightarrow \models_{\mathcal{M}} p(t_1[t/x], \ldots, t_n[t/x])[\sigma] & \text{substitution} \\ & \Leftrightarrow (\bar{\sigma}(t_1[t/x]), \ldots, \bar{\sigma}(t_n[t/x])) \in p^{\mathcal{M}} & \\ & \Leftrightarrow (\overline{\sigma[\bar{\sigma}(t)/x]}(t_1), \ldots, \overline{\sigma[\bar{\sigma}(t)/x]}(t_n)) \in p^{\mathcal{M}} & 1. \\ & \Leftrightarrow \models_{\mathcal{M}} p(t_1, \ldots, t_n)[\sigma[\bar{\sigma}(t)/x]] & \\ & \Leftrightarrow \models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]]. & \end{array}$$

If $\varphi = \bot$, then we have $\not\models_{\mathcal{M}} \varphi[t/x][\sigma]$ and $\not\models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]]$ by definition.

Assume $\varphi = \varphi_1 \wedge \varphi_2$. Then

$$\begin{array}{lll} \models_{\mathcal{M}} \varphi[t/x][\sigma] & \Leftrightarrow \models_{\mathcal{M}} \varphi_1[t/x][\sigma] \text{ and } \models_{\mathcal{M}} \varphi_2[t/x][\sigma] & \\ & \Leftrightarrow \models_{\mathcal{M}} \varphi_1[\sigma[\bar{\sigma}(t)/x]] \text{ and } \models_{\mathcal{M}} \varphi_2[\sigma[\bar{\sigma}(t)/x]] & \text{Ind. Hyp.} \\ & \Leftrightarrow \models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]]. & \end{array}$$

The cases where φ is one of the formulas $\neg\varphi'$, $\varphi_1 \vee \varphi_2$, or $\varphi_1 \to \varphi_2$ are similar to the previous case.

Assume $\varphi = Qy{:}\varphi'$ with $Q \in \{\forall, \exists\}$. In the case $x = y$ the variable x does not occur free in φ so that $\varphi[t/x] = \varphi$. By Lemma 2.2.7(2) we get $\models_{\mathcal{M}} \varphi[\sigma]$ iff $\models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]$, and, hence, the assertion. Assume $x \neq y$. Then we conclude

$$
\begin{aligned}
\models_{\mathcal{M}} \varphi[t/x][\sigma] \Leftrightarrow & \models_{\mathcal{M}} Qy{:}\varphi'[t/x][\sigma] \\
\Leftrightarrow & \models_{\mathcal{M}} \varphi'[t/x][\sigma[a/y]] && \text{for all/some } a \in |\mathcal{M}| \\
\Leftrightarrow & \models_{\mathcal{M}} \varphi'[\sigma[a/y][\overline{\sigma[a/y]}(t)/x]] && \text{for all/some } a \in |\mathcal{M}| \\
\Leftrightarrow & \models_{\mathcal{M}} \varphi'[\sigma[a/y][\bar{\sigma}(t)/x]] && \text{for all/some } a \in |\mathcal{M}| \\
& && \text{by Lemma 2.2.7(2) since } t \\
& && \text{is free for } x \text{ in } Qy{:}\varphi \\
\Leftrightarrow & \models_{\mathcal{M}} \varphi'[\sigma[\bar{\sigma}(t)/x][a/y]] && \text{for all/some } a \in |\mathcal{M}| \\
& && \text{since } x \neq y \\
\Leftrightarrow & \models_{\mathcal{M}} Qy{:}\varphi'[\sigma[\bar{\sigma}(t)/x]] \\
\Leftrightarrow & \models_{\mathcal{M}} \varphi[\sigma[\bar{\sigma}(t)/x]],
\end{aligned}
$$

where the third equivalence is an application of the induction hypothesis. □

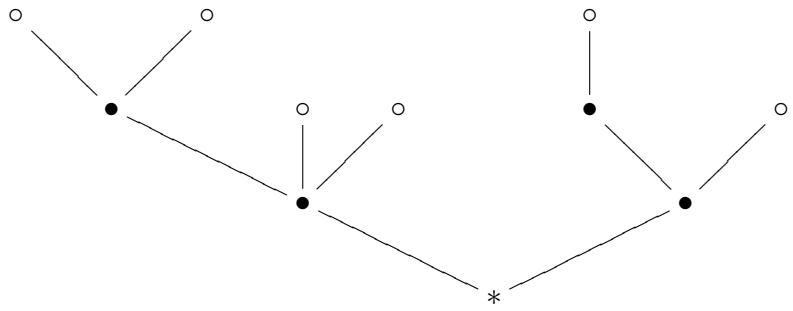
2.3 Natural Deduction

In this section we want to investigate a formal calculus for reasoning about formulas. This calculus, called natural deduction, uses proof rules, which allow to infer formulas from other formulas. By applying these rules in succession, we may infer a conclusion from a finite set of premises. Suppose a set $\{\varphi_1, \ldots, \varphi_n\}$ of formulas is given. We start to apply a proof rule of the calculus to certain elements of the set of premises generating a new formula ψ_1. In the next step we apply a rule to certain elements of the set $\{\varphi_1, \ldots, \varphi_n, \psi_1\}$ generating a new formula ψ_2. Continuous application of the rules to a growing set of formulas will finally end in the intended result ψ - the conclusion. In this case we are successful in deriving ψ from the set of premises, and we will denote that by

$$\varphi_1, \ldots, \varphi_n \vdash \psi.$$

Some rules allow us to make temporary assumptions, i.e., such a rule enlarges the set of premises temporarily. The derivation itself is actually a tree with the premises and temporary assumptions we as leafs, applications of rules as nodes, and the

conclusion as the root. The skeleton of a proof may look as follows ($\circ$ denotes a premises or assumption, $\bullet$ denotes an intermediate formula generated by a certain rule, $*$ denotes the conclusion):



The proof rules of the natural deduction calculus are grouped by the logical operators and constants of the propositional language. For each operator we have introduction and elimination rules, and for the constant $\bot$ a single rule. Introduction rules are used to infer a formula containing the operator as the outermost symbol. Elimination rules are used to derive other properties from a formula containing the operator. We want to discuss these rules in detail.

And introduction: This rule is used to infer a formula of the form $\varphi \wedge \psi$. It seems obvious that we are allowed to conclude this formula if we have already concluded both φ and ψ. Therefore, the rule becomes

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \wedge\text{I}.$$

The rule is binary, i.e, it has to be applied to two subtrees, the first deriving φ, the second deriving ψ. To the right of the line we denote the name of the rule (I means introduction, E means elimination).

And elimination: This rule is used to infer other properties from a formula of the form $\varphi \wedge \psi$. We have to such elimination rules given by

$$\frac{\varphi \wedge \psi}{\varphi} \wedge\text{E1} \qquad \frac{\varphi \wedge \psi}{\psi} \wedge\text{E2}.$$

Or introduction: These rules is used to infer a formula of the form $\varphi \vee \psi$. It seems obvious that is enough to know that either φ or ψ can be derived. Therefore, we obtain the two rules:

$$\frac{\varphi}{\varphi \vee \psi} \vee\text{I1} \qquad \frac{\psi}{\varphi \vee \psi} \vee\text{I2}.$$

Or elimination: We are allowed to conclude a property χ from $\varphi \vee \psi$ if we know that φ as well as ψ implies χ. The or-elimination rule formalizes this principle:

$$\frac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]\\ \vdots\\ \chi\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots\\ \chi\end{array}}{\chi}\ \vee\mathrm{E}.$$

Notice that the middle and right subtree correspond to the proof of χ by assuming φ and ψ, respectively.

Implication introduction: In order to infer a formula of the form $\varphi \to \psi$ we are allowed to temporarily make the assumption φ. From this assumption we have to derive the formula ψ. If we are successful we denote this derivation by

$$\begin{array}{c}\varphi\\ \vdots\\ \psi.\end{array}$$

In that case we are allowed to conclude $\varphi \to \psi$. In addition, the temporary assumption φ is not longer needed. We are allowed to discard it, denoted by $[\varphi]$. The rule finally is:

$$\frac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi \to \psi}\ \to\mathrm{I}.$$

PBC: This rule is neither an introduction nor an elimination rule. PBCis an abbreviation for proof by contradiction. If we are able to show that $\neg\varphi$ leads to a contradiction, the formula $\bot$, then we are allowed to conclude φ. The rule reads:

$$\frac{\begin{array}{c}[\neg\varphi]\\ \vdots\\ \bot\end{array}}{\varphi}\ \mathrm{PBC}.$$

For all elimination: If we know that $\forall x{:}\varphi$ is true, then it is legal to conclude that φ for x being an arbitrary element. In other terms, we are allowed to conclude that $\varphi[t/x]$ is true for an arbitrary term t.

$$\frac{\forall x{:}\varphi}{\varphi[t/x]}\ \forall\mathrm{E}$$

For all introduction: In order to show that a formula $\forall x{:}\varphi$ is true one may simply show φ, i.e., we simply assume that x is an arbitrary element. For this being legal we must assure that the variable x is not already used elsewhere (as a free variable), i.e., that it is a 'fresh/new' variable. We get the rule

$$\frac{\varphi}{\forall x{:}\varphi}\ \forall\mathrm{I} \quad \begin{array}{l}\text{if } x \text{ does not occur free in any}\\ \text{premises of this subtree}\end{array}$$

This rule has condition, which has to be satisfied before we are allowed to apply this rule. Notice that this condition refers to the premises of the subtree, i.e., just to those assumptions that are not yet discarded.

Exists introduction: This rule is very simple. If we were able to derive $\varphi[t/x]$ we have already got a witness t of the existential version of the formula. The rule simply is

$$\frac{\varphi[t/x]}{\exists x{:}\varphi}\ \exists\mathrm{I}$$

Exists elimination: To understand this rule it is helpful to consider the case of a finite model, i.e., a model $\{a_1, \ldots, a_n\}$. In this case an existential formula $\exists x{:}\varphi$ is true if it is true for one of the elements $a_1, \ldots, a_n$. Assume that the terms $t_1, \ldots, t_n$ denote the elements, i.e., $\bar{\sigma}(t_i) = a_i$, then $\exists x{:}\varphi$ is true if $\varphi[t_1/x] \vee \cdots \vee \varphi[t_n/x]$ is true. Consequently, we get a rule similar to $\vee$ elimination. The formula χ must be independent of x, and x be local to that subtree motivating the variable condition of this rule.

$$\frac{\exists x{:}\varphi \qquad \begin{array}{c}[\varphi]\\ \vdots\\ \chi\end{array}}{\chi}\ \exists\mathrm{E} \quad \begin{array}{l}\text{if } x \text{ does not occur free in } \chi \text{ and in any}\\ \text{premises of the right subtree accept } \varphi\end{array}$$

As in the case of $\forall$I the variable condition refers to the premises of the right subtree, i.e., just to those assumptions that are not yet discarded.

The table below lists the rules of natural deduction for first-order logic.

As an example we will provide derivations for several properties in the following two lemmas.

Lemma 2.3.1 *Let $\varphi, \varphi_1, \varphi_2$ be formulas. Then we have:*

1. $\vdash \varphi \leftrightarrow \neg\neg\varphi$.

	introduction rule	elimination rule
$\wedge$	$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\ \wedge\text{I}$	$\dfrac{\varphi \wedge \psi}{\varphi}\ \wedge\text{E1}$ $\quad$ $\dfrac{\varphi \wedge \psi}{\psi}\ \wedge\text{E2}$
$\vee$	$\dfrac{\varphi}{\varphi \vee \psi}\ \vee\text{I1}$ $\quad$ $\dfrac{\psi}{\varphi \vee \psi}\ \vee\text{I2}$	$\dfrac{\varphi \vee \psi \quad \begin{matrix}[\varphi]\\ \vdots\\ \chi\end{matrix} \quad \begin{matrix}[\psi]\\ \vdots\\ \chi\end{matrix}}{\chi}\ \vee\text{E}$
$\rightarrow$	$\dfrac{\begin{matrix}[\varphi]\\ \vdots\\ \psi\end{matrix}}{\varphi \rightarrow \psi}\ \rightarrow\text{I}$	$\dfrac{\varphi \quad \varphi \rightarrow \psi}{\psi}\ \rightarrow\text{E}$
$\neg$	$\dfrac{\begin{matrix}[\varphi]\\ \vdots\\ \bot\end{matrix}}{\neg\varphi}\ \neg\text{I}$	$\dfrac{\varphi \quad \neg\varphi}{\bot}\ \neg\text{E}$
PBC	$\dfrac{\begin{matrix}[\neg\varphi]\\ \vdots\\ \bot\end{matrix}}{\varphi}\ \text{PBC}$	
$\forall$	$\dfrac{\forall x{:}\varphi}{\varphi[t/x]}\ \forall\text{E}$	$\dfrac{\varphi}{\forall x{:}\varphi}\ \forall\text{I}$ if x does not occur free in any premises of this subtree
$\exists$	$\dfrac{\exists x{:}\varphi \quad \begin{matrix}[\varphi]\\ \vdots\\ \chi\end{matrix}}{\chi}\ \exists\text{E}$ if x does not occur free in χ and in any premises of the right subtree accept φ	$\dfrac{\varphi[t/x]}{\exists x{:}\varphi}\ \exists\text{I}$

Table 2.1: Rules of natural deduction for first-order logic

2. $\vdash \neg\varphi_1 \wedge \neg\varphi_2 \leftrightarrow \neg(\varphi_1 \vee \varphi_2)$.

3. $\vdash \neg\varphi_1 \vee \neg\varphi_2 \leftrightarrow \neg(\varphi_1 \wedge \varphi_2)$.

4. $\vdash \varphi \vee \neg\varphi$.

5. $\vdash (\varphi_1 \to \varphi_2) \leftrightarrow \neg\varphi_1 \vee \varphi_2$.

6. $\vdash \neg(\varphi_1 \to \varphi_2) \leftrightarrow \varphi_1 \wedge \neg\varphi_2$.

Proof. In each case we give derivations for both implications.

1.

$$\dfrac{\dfrac{\dfrac{[\neg\varphi]^1 \quad [\varphi]^2}{\bot}\,\neg\mathrm{E}}{\neg\neg\varphi}\,\neg\mathrm{I}^1}{\varphi \to \neg\neg\varphi}\,\to\mathrm{I}^2 \qquad \dfrac{\dfrac{\dfrac{[\neg\neg\varphi]^2 \quad [\neg\varphi]^1}{\bot}\,\neg\mathrm{E}}{\varphi}\,\mathrm{PBC}^1}{\neg\neg\varphi \to \varphi}\,\to\mathrm{I}^2$$

2.

$$\dfrac{\dfrac{\dfrac{[\varphi_1 \vee \varphi_2]^2 \qquad \dfrac{\dfrac{[\neg\varphi_1 \wedge \neg\varphi_2]^3}{\neg\varphi_1}\,\wedge\mathrm{E1} \quad [\varphi_1]^1}{\bot}\,\neg\mathrm{E} \qquad \dfrac{\dfrac{[\neg\varphi_1 \wedge \neg\varphi_2]^3}{\neg\varphi_2}\,\wedge\mathrm{E2} \quad [\varphi_2]^1}{\bot}\,\neg\mathrm{E}}{\bot}\,\vee\mathrm{E}^1}{\neg(\varphi_1 \vee \varphi_2)}\,\neg\mathrm{I}^2}{\neg\varphi_1 \wedge \neg\varphi_2 \to \neg(\varphi_1 \vee \varphi_2)}\,\to\mathrm{I}^3$$

$$\dfrac{\dfrac{\dfrac{[\neg(\varphi_1 \vee \varphi_2)]^3 \quad \dfrac{[\varphi_1]^1}{\varphi_1 \vee \varphi_2}\,\vee\mathrm{I1}}{\bot}\,\neg\mathrm{E} \qquad \dfrac{[\neg(\varphi_1 \vee \varphi_2)]^3 \quad \dfrac{[\varphi_2]^2}{\varphi_1 \vee \varphi_2}\,\vee\mathrm{I2}}{\bot}\,\neg\mathrm{E}}{\neg\varphi_1}\,\neg\mathrm{I}^1 \qquad \dfrac{}{\neg\varphi_2}\,\neg\mathrm{I}^2}{\neg\varphi_1 \wedge \neg\varphi_2}\,\wedge\mathrm{I}}{\neg(\varphi_1 \vee \varphi_2) \to \neg\varphi_1 \wedge \neg\varphi_2}\,\to\mathrm{I}^3$$

3.

$$\dfrac{\dfrac{\dfrac{[\neg\varphi_1 \vee \neg\varphi_2]^3 \qquad \dfrac{\dfrac{[\neg\varphi_1]^1 \quad \dfrac{[\varphi_1 \wedge \varphi_2]^2}{\varphi_1}\,\wedge\mathrm{E1}}{\bot}\,\neg\mathrm{E} \qquad \dfrac{[\neg\varphi_2]^1 \quad \dfrac{[\varphi_1 \wedge \varphi_2]^2}{\varphi_2}\,\wedge\mathrm{E1}}{\bot}\,\neg\mathrm{E}}{\bot}\,\vee\mathrm{E}^1}{\neg(\varphi_1 \wedge \varphi_2)}\,\neg\mathrm{I}^2}{\neg\varphi_1 \vee \neg\varphi_2 \to \neg(\varphi_1 \wedge \varphi_2)}\,\to\mathrm{I}^3$$

For the second derivation we first show that $\neg(\neg\varphi_1 \vee \neg\varphi_2) \vdash \varphi_i$ for $i = 1, 2$.

$$\dfrac{\dfrac{\neg(\neg\varphi_1 \vee \neg\varphi_2) \quad \dfrac{[\neg\varphi_i]^1}{\neg\varphi_1 \vee \neg\varphi_2}\,\vee\text{Ii}}{\bot}\,\neg\text{E}}{\varphi_i}\,\text{PBC}^1$$

Using the derivation above we get

$$\dfrac{\dfrac{[\neg(\varphi_1 \wedge \varphi_2)]^2 \qquad \dfrac{\dfrac{[\neg(\neg\varphi_1 \vee \neg\varphi_2)]^1}{\vdots}}{\varphi_1} \quad \dfrac{\dfrac{[\neg(\neg\varphi_1 \vee \neg\varphi_2)]^1}{\vdots}}{\varphi_2}\,\wedge\text{I}}{\varphi_1 \wedge \varphi_2}\,\neg\text{E}}{\bot}\,\text{PBC}^1}{\neg\varphi_1 \vee \neg\varphi_2}\,\to\text{I}^2$$

$$\neg(\varphi_1 \wedge \varphi_2) \to \neg\varphi_1 \vee \neg\varphi_2$$

4.

$$\dfrac{\dfrac{[\neg(\varphi \vee \neg\varphi)]^2 \qquad \dfrac{\dfrac{[\neg(\varphi \vee \neg\varphi)]^2 \qquad \dfrac{\dfrac{[\varphi]^1}{\varphi \vee \neg\varphi}\,\vee\text{I1}}{\bot}\,\neg\text{E}}{\neg\varphi}\,\text{PBC}^1}{\varphi \vee \neg\varphi}\,\vee\text{I2}}{\bot}\,\neg\text{E}}{\varphi \vee \neg\varphi}\,\text{PBC}^2$$

5. The first derivation uses 4.

$$\dfrac{\varphi_1 \vee \neg\varphi_1 \qquad \dfrac{\dfrac{[\varphi_1 \to \varphi_2]^2 \quad [\varphi_1]^1}{\varphi_2}\,\to\text{E}}{\neg\varphi_1 \vee \varphi_2}\,\vee\text{I2} \qquad \dfrac{[\neg\varphi_1]^1}{\neg\varphi_1 \vee \varphi_2}\,\vee\text{I1}}{\neg\varphi_1 \vee \varphi_2}\,\vee\text{E}^1}{(\varphi_1 \to \varphi_2) \to \neg\varphi_1 \vee \varphi_2}\,\to\text{I}^2$$

$$\dfrac{\dfrac{[\neg\varphi_1 \vee \varphi_2]^3 \qquad \dfrac{\dfrac{[\neg\varphi_1]^1 \quad [\varphi_1]^2}{\bot}\,\neg\text{E}}{\varphi_2}\,\text{PBC} \qquad [\varphi_2]^1}{\varphi_1 \to \varphi_2}\,\to\text{I}^2}{\neg\varphi_1 \vee \varphi_2}\,\vee\text{E}^1}{\neg\varphi_1 \vee \varphi_2 \to (\varphi_1 \to \varphi_2)}\,\to\text{I}^3$$

6.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\neg\varphi_1]^2 \quad [\varphi_1]^1}{\bot}\neg\mathrm{E}}{\varphi_2}\mathrm{PBC}}{\varphi_1 \to \varphi_2}\to\mathrm{I}^1 \quad \dfrac{[\neg(\varphi_1 \to \varphi_2)]^4 \quad \dfrac{[\varphi_2]^3}{\varphi_1 \to \varphi_2}\to\mathrm{I}}{\bot}\neg\mathrm{E}}{\neg\varphi_2}\neg\mathrm{I}^3}{\varphi_1 \wedge \neg\varphi_2}\wedge\mathrm{I}}{\neg(\varphi_1 \to \varphi_2) \to \varphi_1 \wedge \neg\varphi_2}\to\mathrm{I}^4$$

$$\dfrac{\dfrac{\dfrac{[\varphi_1 \wedge \neg\varphi_2]^2}{\neg\varphi_2}\wedge\mathrm{E2} \quad \dfrac{[\varphi_1 \to \varphi_2]^1 \quad \dfrac{[\varphi_1 \wedge \neg\varphi_2]^2}{\varphi_1}\wedge\mathrm{E1}}{\varphi_2}\to\mathrm{E}}{\bot}\neg\mathrm{E}}{\neg(\varphi_1 \to \varphi_2)}\neg\mathrm{I}^1}{\varphi_1 \wedge \neg\varphi_2 \to \neg(\varphi_1 \to \varphi_2)}\to\mathrm{I}^2$$

This completes the proof. □

Lemma 2.3.2 *Let $\varphi, \varphi_1, \varphi_2, \varphi_3 \in$ be formulas. Then we have:*

1. $\vdash \varphi_1 \wedge \varphi_2 \leftrightarrow \varphi_2 \wedge \varphi_1$.
2. $\vdash \varphi_1 \vee \varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1$.
3. $\vdash \bot \leftrightarrow \varphi \wedge \neg\varphi$.
4. $\vdash \varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \leftrightarrow (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$.
5. $\vdash \varphi_1 \vee (\varphi_2 \vee \varphi_3) \leftrightarrow (\varphi_1 \vee \varphi_2) \vee \varphi_3$.
6. $\vdash \varphi_1 \wedge (\varphi_2 \vee \varphi_3) \leftrightarrow (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$.
7. $\vdash \varphi_1 \vee (\varphi_2 \wedge \varphi_3) \leftrightarrow (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$.

Proof. We provide derivations for both implications.

1.

$$\dfrac{\dfrac{\dfrac{[\varphi_1 \wedge \varphi_2]^1}{\varphi_2}\wedge\mathrm{E2} \quad \dfrac{[\varphi_1 \wedge \varphi_2]^1}{\varphi_1}\wedge\mathrm{E1}}{\varphi_2 \wedge \varphi_1}\wedge\mathrm{I}}{\varphi_1 \wedge \varphi_2 \to \varphi_2 \wedge \varphi_1}\to\mathrm{I}^1$$

The second derivation is similar.

2.

$$\dfrac{\dfrac{[\varphi_1 \vee \varphi_2]^2 \quad \dfrac{[\varphi_1]^1}{\varphi_2 \vee \varphi_1}\vee\text{I2} \quad \dfrac{[\varphi_2]^1}{\varphi_2 \vee \varphi_1}\vee\text{I1}}{\varphi_2 \vee \varphi_1}\vee\text{E}^1}{\varphi_1 \vee \varphi_2 \to \varphi_2 \vee \varphi_1}\to\text{I}^2$$

The second derivation is similar.

3.

$$\dfrac{\dfrac{[\bot]^1}{\varphi \wedge \neg\varphi}\,\text{PBC}}{\bot \to \varphi \wedge \neg\varphi}\to\text{I}^1 \qquad \dfrac{\dfrac{\dfrac{[\varphi \wedge \neg\varphi]^1}{\neg\varphi}\wedge\text{E2} \quad \dfrac{[\varphi \wedge \neg\varphi]^1}{\varphi}\wedge\text{E1}}{\bot}\neg\text{E}}{\varphi \wedge \neg\varphi \to \bot}\to\text{I}^1$$

4.

$$\dfrac{\dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)]^1}{\varphi_1}\wedge\text{E1} \quad \dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)]^1}{\varphi_2 \wedge \varphi_3}\wedge\text{E2}}{\varphi_2}\wedge\text{E1}}{\varphi_1 \wedge \varphi_2}\wedge\text{I} \quad \dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)]^1}{\varphi_2 \wedge \varphi_3}\wedge\text{E2}}{\varphi_3}\wedge\text{E2}}{(\varphi_1 \wedge \varphi_2) \wedge \varphi_3}\wedge\text{I}}{\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \to (\varphi_1 \wedge \varphi_2) \wedge \varphi_3}\to\text{I}^1$$

The second derivation is similar.

5.

$$\dfrac{\dfrac{[\varphi_1 \vee (\varphi_2 \vee \varphi_3)]^3 \quad \dfrac{\dfrac{[\varphi_1]^2}{\varphi_1 \vee \varphi_2}\vee\mathrm{I1}}{(\varphi_1 \vee \varphi_2) \vee \varphi_3}\vee\mathrm{I1} \quad \dfrac{[\varphi_2 \vee \varphi_3]^2 \quad \dfrac{\dfrac{[\varphi_2]^1}{\varphi_1 \vee \varphi_2}\vee\mathrm{I2}}{(\varphi_1 \vee \varphi_2) \vee \varphi_3}\vee\mathrm{I1} \quad \dfrac{[\varphi_3]^1}{(\varphi_1 \vee \varphi_2) \vee \varphi_3}\vee\mathrm{I2}}{(\varphi_1 \vee \varphi_2) \vee \varphi_3}\vee\mathrm{E}^1}{(\varphi_1 \vee \varphi_2) \vee \varphi_3}\vee\mathrm{E}^2}{\varphi_1 \vee (\varphi_2 \vee \varphi_3) \to (\varphi_1 \vee \varphi_2) \vee \varphi_3}\to\mathrm{I}^3$$

The second derivation is similar.

6.

$$\dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \vee \varphi_3)]^2}{\varphi_2 \vee \varphi_3}\wedge\mathrm{E2} \quad \dfrac{\dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \vee \varphi_3)]^2}{\varphi_1}\wedge\mathrm{E1} \quad \dfrac{[\varphi_2]^1}{}}{\varphi_1 \wedge \varphi_2}\wedge\mathrm{I}}{(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)}\vee\mathrm{I1} \quad \dfrac{\dfrac{\dfrac{[\varphi_1 \wedge (\varphi_2 \vee \varphi_3)]^2}{\varphi_1}\wedge\mathrm{E1} \quad \dfrac{[\varphi_3]^1}{}}{\varphi_1 \wedge \varphi_3}\wedge\mathrm{I}}{(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)}\vee\mathrm{I2}}{(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)}\vee\mathrm{E}^1}{\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \to (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)}\to\mathrm{I}^2$$

$$\dfrac{\dfrac{[(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)]^2 \quad \dfrac{\dfrac{\dfrac{[\varphi_1 \wedge \varphi_2]^1}{\varphi_1}\wedge\mathrm{E1} \quad \dfrac{[\varphi_1 \wedge \varphi_2]^1}{\varphi_2}\wedge\mathrm{E2}}{\varphi_2 \wedge \varphi_3}\vee\mathrm{I1}}{\varphi_1 \wedge (\varphi_2 \vee \varphi_3)}\wedge\mathrm{I} \quad \dfrac{\dfrac{\dfrac{[\varphi_1 \wedge \varphi_3]^1}{\varphi_1}\wedge\mathrm{E1} \quad \dfrac{[\varphi_1 \wedge \varphi_3]^1}{\varphi_3}\wedge\mathrm{E2}}{\varphi_2 \wedge \varphi_3}\vee\mathrm{I2}}{\varphi_1 \wedge (\varphi_2 \vee \varphi_3)}\wedge\mathrm{I}}{\varphi_1 \wedge (\varphi_2 \vee \varphi_3)}\vee\mathrm{E}^1}{(\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3) \to \varphi_1 \wedge (\varphi_2 \vee \varphi_3)}\to\mathrm{I}^2$$

7.

$$\dfrac{[\varphi_1 \vee (\varphi_2 \wedge \varphi_3)]^2 \quad \dfrac{\dfrac{[\varphi_1]^1}{\varphi_1 \vee \varphi_2}\vee\mathrm{I1} \quad \dfrac{[\varphi_1]^1}{\varphi_1 \vee \varphi_3}\vee\mathrm{I1}}{(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}\wedge\mathrm{I} \quad \dfrac{\dfrac{\dfrac{[\varphi_2 \wedge \varphi_3]^1}{\varphi_2}\wedge\mathrm{E1}}{\varphi_1 \vee \varphi_2}\vee\mathrm{I2} \quad \dfrac{\dfrac{[\varphi_2 \wedge \varphi_3]^1}{\varphi_3}\wedge\mathrm{E2}}{\varphi_1 \vee \varphi_3}\vee\mathrm{I2}}{(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}\wedge\mathrm{I}}{(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}\vee\mathrm{E}^1}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \to (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}\to\mathrm{I}^2$$

For the second derivation we first give a derivation $(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3), \varphi_3 \vdash \varphi_1 \vee (\varphi_2 \wedge \varphi_3)$.

$$\dfrac{\dfrac{(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}{\varphi_1 \vee \varphi_2}\wedge\mathrm{E1} \quad \dfrac{[\varphi_1]^1}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\vee\mathrm{I1} \quad \dfrac{\dfrac{[\varphi_2]^1 \quad \varphi_3}{\varphi_2 \wedge \varphi_3}\wedge\mathrm{I}}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\vee\mathrm{I2}}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\vee\mathrm{E}^1$$

Using the result above we get the following.

$$\dfrac{\dfrac{[(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)]^2}{\varphi_1 \vee \varphi_3}\wedge\mathrm{E2} \quad \dfrac{[\varphi_1]^1}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\vee\mathrm{I1} \quad \begin{array}{c}[(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)]^2 \quad [\varphi_3]^1 \\ \vdots \\ \varphi_1 \vee (\varphi_2 \wedge \varphi_3)\end{array}}{\varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\vee\mathrm{E}^1}{(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \to \varphi_1 \vee (\varphi_2 \wedge \varphi_3)}\to\mathrm{I}^2$$

As another example using the rules for quantification we provide again two lemmas.

Lemma 2.3.3 *Let $\varphi \in$ FOL be a formula not containing a free occurrence of the variable y. The we have*

1. $\vdash \forall x{:}\varphi \leftrightarrow \forall y{:}\varphi[y/x]$.
2. $\vdash \exists x{:}\varphi \leftrightarrow \exists y{:}\varphi[y/x]$.

Proof. In both cases it is sufficient to show '$\rightarrow$'. The other implication is similar.

1.
$$\dfrac{\dfrac{\dfrac{[\forall x{:}\varphi]^1}{\varphi[y/x]}\,\forall\mathrm{E}}{\forall y{:}\varphi[y/x]}\,\forall\mathrm{I}}{\forall x{:}\varphi \rightarrow \forall y{:}\varphi[y/x]}\,{\rightarrow}\mathrm{I}^1$$
Notice that the variable condition of $\forall$I is satisfied since φ, and, hence, $\forall x{:}\varphi$ does not contain a free occurrence of y.

2.
$$\dfrac{\dfrac{[\exists x{:}\varphi]^2 \quad \dfrac{[\varphi]^1}{\exists y{:}\varphi[y/x]}\,\exists\mathrm{I}}{\exists y{:}\varphi[y/x]}\,\exists\mathrm{E}^1}{\exists x{:}\varphi \rightarrow \exists y{:}\varphi[y/x]}\,{\rightarrow}\mathrm{I}^2$$
Notice that the $\exists$I is of the required form since $\varphi[y/x][x/y]$ is just φ (recall y does not occur free in φ). Furthermore, the variable condition of $\exists$E is satisfied since x does not occur free in $\exists y{:}\varphi[y/x]$. It occurs free in φ, which does not violate the condition.

This completes the proof. □

Lemma 2.3.4 *Let $\varphi, \varphi_1, \varphi_2, \varphi_3 \in$ FOL be formulas so that x does not occur free in φ_3. Then we have:*

1. $\vdash \forall x{:}\forall y{:}\varphi \leftrightarrow \forall y{:}\forall x{:}\varphi$.
2. $\vdash \forall x{:}\forall y{:}\varphi \leftrightarrow \forall y{:}\forall x{:}\varphi$.
3. $\vdash \neg\forall x{:}\varphi \leftrightarrow \exists x{:}\neg\varphi$.

4. $\vdash \neg\exists x{:}\varphi \leftrightarrow \forall x{:}\neg\varphi$.

5. $\vdash \forall x{:}(\varphi_1 \wedge \varphi_3) \leftrightarrow \forall x{:}\varphi_1 \wedge \varphi_3$.

6. $\vdash \forall x{:}(\varphi_1 \wedge \varphi_2) \leftrightarrow \forall x{:}\varphi_1 \wedge \forall x{:}\varphi_2$.

7. $\vdash \forall x{:}(\varphi_1 \vee \varphi_3) \leftrightarrow \forall x{:}\varphi_1 \vee \varphi_3$.

8. $\vdash \exists x{:}(\varphi_1 \vee \varphi_3) \leftrightarrow \exists x{:}\varphi_1 \vee \varphi_3$.

9. $\vdash \exists x{:}(\varphi_1 \vee \varphi_2) \leftrightarrow \exists x{:}\varphi_1 \vee \exists x{:}\varphi_2$.

10. $\vdash \exists x{:}(\varphi_1 \wedge \varphi_3) \leftrightarrow \exists x{:}\varphi_1 \wedge \varphi_3$.

Proof. In the following derivation we will always omit the application of $\rightarrow$I and $\leftrightarrow$I.

1. We just prove $\rightarrow$ since the converse implication follows analogously:

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall x{:}\forall y{:}\varphi}{\forall y{:}\varphi}\,\forall\mathrm{E}}{\varphi}\,\forall\mathrm{E}}{\forall x{:}\varphi}\,\forall\mathrm{I}}{\forall y{:}\forall x{:}\varphi}\,\forall\mathrm{I}$$

   The variable condition in the two applications is satisfied since $\forall x{:}\forall y{:}\varphi$ does not contain x or y freely.

2. Again, we just prove $\rightarrow$:

$$\dfrac{\exists x{:}\exists y{:}\varphi \qquad \dfrac{[\exists y{:}\varphi]^2 \qquad \dfrac{\dfrac{[\varphi]^1}{\exists x{:}\varphi}\,\exists\mathrm{I}}{\exists y{:}\exists x{:}\varphi}\,\exists\mathrm{I}}{\exists y{:}\exists x{:}\varphi}\,\exists\mathrm{E}^1}{\exists y{:}\exists x{:}\varphi}\,\exists\mathrm{E}^2$$

   The variable condition in the two applications of $\exists$E is satisfied since $\exists y{:}\exists x{:}\varphi$ does not contain x or y freely.

3.

$$\dfrac{\dfrac{\dfrac{[\neg\exists x{:}\neg\varphi]^2 \quad \dfrac{[\neg\varphi]^1}{\exists x{:}\neg\varphi}\,\exists\mathrm{I}}{\dfrac{\bot}{\varphi}\,\mathrm{PBC}^1}\,\neg\mathrm{E}}{\forall x{:}\varphi}\,\forall\mathrm{I} \quad \neg\forall x{:}\varphi}{\dfrac{\bot}{\exists x{:}\neg\varphi}\,\mathrm{PBC}^2}\,\neg\mathrm{E} \qquad \dfrac{\dfrac{\exists x{:}\neg\varphi \quad \dfrac{\dfrac{[\neg\varphi]^1 \quad \dfrac{[\forall x{:}\varphi]^2}{\varphi}\,\forall\mathrm{E}}{\bot}\,\neg\mathrm{E}}{\bot}\,\exists\mathrm{E}^1}{\neg\forall x{:}\varphi}\,\neg\mathrm{I}^2$$

The variable conditions of ∀I in the first derivation and of ∃E in the second derivation are satisfied since x does neither occur free in $\neg\exists x{:}\neg\varphi$ nor in $\bot$ and $\forall x{:}\varphi$.

4.

$$\dfrac{\dfrac{\neg\exists x{:}\varphi \quad \dfrac{[\varphi]^1}{\exists x{:}\varphi}\,\exists\mathrm{I}}{\dfrac{\bot}{\neg\varphi}\,\neg\mathrm{I}^1}\,\neg\mathrm{E}}{\forall x{:}\neg\varphi}\,\forall\mathrm{I} \qquad \dfrac{\dfrac{[\exists x{:}\varphi]^2 \quad \dfrac{\dfrac{\forall x{:}\neg\varphi}{\neg\varphi}\,\forall\mathrm{E} \quad [\varphi]^1}{\bot}\,\neg\mathrm{E}}{\bot}\,\exists\mathrm{E}^1}{\neg\exists x{:}\varphi}\,\neg\mathrm{I}^2$$

The variable conditions of ∀I in the first derivation (provided by Shahid Mahmood) and of ∃E in the second derivation are satisfied since x does neither occur free in $\neg\exists x{:}\varphi$ nor in $\bot$ and $\forall x{:}\neg\varphi$.

5.

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall x{:}(\varphi_1\wedge\varphi_3)}{\varphi_1\wedge\varphi_3}\,\forall\mathrm{E}}{\varphi_1}\,\wedge\mathrm{E1}}{\forall x{:}\varphi_1}\,\forall\mathrm{I} \quad \dfrac{\dfrac{\forall x{:}(\varphi_1\wedge\varphi_3)}{\varphi_1\wedge\varphi_3}\,\forall\mathrm{E}}{\varphi_3}\,\wedge\mathrm{E2}}{\forall x{:}\varphi_1\wedge\varphi_3}\,\wedge\mathrm{I} \qquad \dfrac{\dfrac{\dfrac{\forall x{:}\varphi_1\wedge\varphi_3}{\forall x{:}\varphi_1}\,\wedge\mathrm{E1}}{\varphi_1}\,\forall\mathrm{E} \quad \dfrac{\forall x{:}\varphi_1\wedge\varphi_3}{\varphi_3}\,\wedge\mathrm{E2}}{\varphi_1\wedge\varphi_3}\,\wedge\mathrm{I}}{\forall x{:}(\varphi_1\wedge\varphi_3)}\,\forall\mathrm{I}$$

The variable conditions of ∀I in the first derivation and the second derivation are satisfied since x does neither occur free in $\forall x{:}(\varphi_1\wedge\varphi_3)$ nor in $\forall x{:}\varphi_1\wedge\varphi_3$ (assumption on φ_3).

6. This derivation is similar to the previous one by adding an application of ∀I, respectively of ∀E, in the second branch of both parts.

7. For the implication $\rightarrow$ we first provide a derivation $\forall x{:}(\varphi_1\vee\varphi_3), \neg\varphi_1 \vdash \forall x{:}\varphi_1\vee\varphi_3$:

$$\dfrac{\dfrac{\forall x{:}(\varphi_1\vee\varphi_3)}{\varphi_1\vee\varphi_3}\,\forall\mathrm{E} \quad \dfrac{\dfrac{\neg\varphi_1 \quad [\varphi_1]^1}{\bot}\,\neg\mathrm{E}}{\forall x{:}\varphi_1\vee\varphi_3}\,\mathrm{PBC} \quad \dfrac{[\varphi_3]^1}{\forall x{:}\varphi_1\vee\varphi_3}\,\vee\mathrm{I2}}{\forall x{:}\varphi_1\vee\varphi_3}\,\vee\mathrm{E}^1$$

Using the derivation above we get $\forall x{:}(\varphi_1 \vee \varphi_3), \neg\forall x{:}\varphi_1 \vdash \forall x{:}\varphi_1 \vee \varphi_3$ as follows:

$$\dfrac{\dfrac{\begin{array}{c}(3)\\ \vdots \\ \neg\forall x{:}\varphi_1 \to \exists x{:}\neg\varphi_1\end{array} \quad \neg\forall x{:}\varphi_1}{\exists x{:}\neg\varphi_1}\,{\to}\mathrm{E} \qquad \begin{array}{c}\forall x{:}(\varphi_1 \vee \varphi_3)\cdots[\neg\varphi_1]^1\\ \vdots \\ \forall x{:}\varphi_1 \vee \varphi_3\end{array}}{\forall x{:}\varphi_1 \vee \varphi_3}\,\exists\mathrm{E}^1$$

The variable condition for $\exists$E is satisfied since x does neither occur free in $\forall x{:}(\varphi_1 \vee \varphi_3)$ nor in $\forall x{:}\varphi_1 \vee \varphi_3$ (assumption on φ_3). Finally, using the derivation above we get

$$\dfrac{\begin{array}{c}\text{Lemma 2.3.1(4)}\\ \vdots \\ \forall x{:}\varphi_1 \vee \neg\forall x{:}\varphi_1\end{array} \quad \dfrac{[\forall x{:}\varphi_1]^1}{\forall x{:}\varphi_1 \vee \varphi_3}\,\vee\mathrm{I1} \qquad \begin{array}{c}\forall x{:}(\varphi_1 \vee \varphi_3)\ldots[\neg\forall x{:}\varphi_1]^1\\ \vdots \\ \forall x{:}\varphi_1 \vee \varphi_3\end{array}}{\forall x{:}\varphi_1 \vee \varphi_3}\,\vee\mathrm{E}^1$$

The converse implication follows from

$$\dfrac{\forall x{:}\varphi_1 \vee \varphi_3 \quad \dfrac{\dfrac{\dfrac{[\forall x{:}\varphi_1]^1}{\varphi_1}\,\forall\mathrm{E}}{\varphi_1 \vee \varphi_3}\,\vee\mathrm{I1} \quad \dfrac{[\varphi_3]^1}{\varphi_1 \vee \varphi_3}\,\vee\mathrm{I2}}{\varphi_1 \vee \varphi_3}\,\vee\mathrm{E}^1}{\forall x{:}(\varphi_1 \vee \varphi_3)}\,\forall\mathrm{I}$$

The variable condition of $\forall$I is satisfied since x does not occur free in $\forall x{:}\varphi_1 \vee \varphi_3$ (assumption on φ_3).

8.

$$\dfrac{\exists x{:}(\varphi_1 \vee \varphi_3) \quad \dfrac{[\varphi_1 \vee \varphi_3]^2 \quad \dfrac{\dfrac{[\varphi_1]^2}{\exists x{:}\varphi_1}\,\exists\mathrm{I}}{\exists x{:}\varphi_1 \vee \varphi_3}\,\vee\mathrm{I1} \quad \dfrac{[\varphi_3]^2}{\exists x{:}\varphi_1 \vee \varphi_3}\,\vee\mathrm{I2}}{\exists x{:}\varphi_1 \vee \varphi_3}\,\vee\mathrm{E}^1}{\exists x{:}\varphi_1 \vee \varphi_3}\,\exists\mathrm{E}^2$$

The variable condition of $\exists$E is satisfied since x does not occur free in $\exists x{:}\varphi_1 \vee \varphi_3$ (assumption on φ_3).

$$\dfrac{\exists x{:}\varphi_1 \vee \varphi_3 \quad \dfrac{[\exists x{:}\varphi_1]^2 \quad \dfrac{\dfrac{[\varphi_1]^1}{\varphi_1 \vee \varphi_3}\,\vee\mathrm{I1}}{\exists x{:}(\varphi_1 \vee \varphi_3)}\,\exists\mathrm{I}}{\exists x{:}(\varphi_1 \vee \varphi_3)}\,\exists\mathrm{E}^1 \quad \dfrac{\dfrac{[\varphi_3]^2}{\varphi_1 \vee \varphi_3}\,\vee\mathrm{I2}}{\exists x{:}(\varphi_1 \vee \varphi_3)}\,\exists\mathrm{I}}{\exists x{:}(\varphi_1 \vee \varphi_3)}\,\vee\mathrm{E}^2$$

The variable condition of $\exists$E is satisfied since x does not occur free in $\exists x{:}(\varphi_1 \vee \varphi_3)$.

9. This derivation is similar to the previous one by adding an application of $\exists$I, respectively of $\exists$E, in the right most branch of both parts.

10.
$$\dfrac{\exists x{:}(\varphi_1 \wedge \varphi_3) \qquad \dfrac{\dfrac{\dfrac{[\varphi_1 \wedge \varphi_3]^1}{\varphi_1}\wedge\text{E1}}{\exists x{:}\varphi_1}\exists\text{I} \qquad \dfrac{[\varphi_1 \wedge \varphi_3]^1}{\varphi_3}\wedge\text{E2}}{\exists x{:}\varphi_1 \wedge \varphi_3}\wedge\text{I}}{\exists x{:}\varphi_1 \wedge \varphi_3}\exists\text{E}^1$$

The variable condition of $\exists$E is satisfied since x does not occur free in $\exists x{:}\varphi_1 \wedge \varphi_3$ (assumption on φ_3).

$$\dfrac{\dfrac{\exists x{:}\varphi_1 \wedge \varphi_3}{\exists x{:}\varphi_1}\wedge\text{E1} \qquad \dfrac{\dfrac{[\varphi_1]^1 \qquad \dfrac{\exists x{:}\varphi_1 \wedge \varphi_3}{\varphi_3}\wedge\text{E2}}{\varphi_1 \wedge \varphi_3}\wedge\text{I}}{\exists x{:}(\varphi_1 \wedge \varphi_3)}\exists\text{I}}{\exists x{:}(\varphi_1 \wedge \varphi_3)}\exists\text{E}^1$$

The variable condition of $\exists$E is satisfied since x does neither occur free in $\exists x{:}(\varphi_1 \wedge \varphi_3)$ nor in $\exists x{:}\varphi_1 \wedge \varphi_3$ (assumption on φ_3). □

We will provide further derivation when we consider normal forms of formulas in first-order logic.

Theorem 2.3.5 (Soundness) *Let $\varphi_1, \ldots, \varphi_n$ and ψ be formulas. If $\varphi_1, \ldots, \varphi_n \vdash \psi$, then $\varphi_1, \ldots, \varphi_n \models \psi$ holds.*

Proof. The proof is done by induction on the derivation $\varphi_1, \ldots, \varphi_n \vdash \psi$.

(Base case): In this case the proof is just a premises, i.e., we have $\psi \in \{\varphi_1, \ldots, \varphi_n\}$. Assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. Then we conclude

$$\models_{\mathcal{M}} \psi[\sigma]$$

, and, hence $\varphi_1, \ldots, \varphi_n \models \psi$.

(Induction step): We distinguish several cases according the last rule applied.

$\wedge$**I** : In this case $\psi = \psi_1 \wedge \psi_2$ and we have derivations $\varphi_1, \ldots, \varphi_n \vdash \psi_1$ and $\varphi_1, \ldots, \varphi_n \vdash \psi_2$. From the induction hypothesis we get $\varphi_1, \ldots, \varphi_n \models \psi_1$ and $\varphi_1, \ldots, \varphi_n \models \psi_2$. Now, assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. Then we obtain from the induction hypothesis that $\models_{\mathcal{M}} \psi_1[\sigma]$ and $\models_{\mathcal{M}} \psi_2[\sigma]$. We conclude $\models_{\mathcal{M}} \psi[\sigma]$, and, hence $\varphi_1, \ldots, \varphi_n \models \psi$.

$\wedge$**E1** : In this case we have a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi \wedge \psi'$ for some formula ψ'. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises, i.e., $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. By the induction hypothesis we conclude $\models_{\mathcal{M}} \psi \wedge \psi'[\sigma]$, which implies $\models_{\mathcal{M}} \psi[\sigma]$, and, hence, $\varphi_1, \ldots, \varphi_n \models \psi$.

$\wedge$**E2** : Analogously to $\wedge$E1.

$\vee$**I1** : In this case $\psi = \psi_1 \vee \psi_2$ and we have a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi_1$. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises. By the induction hypothesis we conclude $\models_{\mathcal{M}} \psi_1[\sigma]$, which implies $\models_{\mathcal{M}} \psi[\sigma]$, and, hence, $\varphi_1, \ldots, \varphi_n \models \psi$.

$\vee$**I2** : Analogously to $\vee$I1.

$\vee$**E** : In this case we have the following derivations

$$\begin{array}{ll} \varphi_1, \ldots, \varphi_n & \vdash \psi_1 \vee \psi_2 \\ \varphi_1, \ldots, \varphi_n, \psi_1 & \vdash \psi \\ \varphi_1, \ldots, \varphi_n, \psi_2 & \vdash \psi \end{array}$$

Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises. We get $\models_{\mathcal{M}} \psi_1 \vee \psi_2[\sigma]$ from the induction hypothesis, i.e., either $\models_{\mathcal{M}} \psi_1[\sigma]$ or $\models_{\mathcal{M}} \psi_2[\sigma]$. In the first case we conclude that $\mathcal{M}$ and σ satisfy $\{\varphi_1, \ldots, \varphi_n, \psi_1\}$. Using the induction hypothesis again we get $\models_{\mathcal{M}} \psi[\sigma]$. If $\models_{\mathcal{M}} \psi_2[\sigma]$ we conclude $\models_{\mathcal{M}} \psi[\sigma]$ analogously.

$\rightarrow$**I** : In this case $\psi = \psi_1 \rightarrow \psi_2$ and we have a derivation $\varphi_1, \ldots, \varphi_n, \psi_1 \vdash \psi_2$. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises. If $\mathcal{M}$ and σ also satisfy ψ_1 we conclude $\models_{\mathcal{M}} \psi_2[\sigma]$ from the induction hypothesis, and, hence, $\models_{\mathcal{M}} \psi[\sigma]$. If $\mathcal{M}$ and σ do not satisfy ψ_1 we immediately get $\models_{\mathcal{M}} \psi[\sigma]$.

$\rightarrow$**E** : In this case we have derivations $\varphi_1, \ldots, \varphi_n \vdash \psi'$ and $\varphi_1, \ldots, \varphi_n \vdash \psi' \rightarrow \psi$ for some formula ψ'. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises. From the induction hypothesis we get $\models_{\mathcal{M}} \psi'[\sigma]$ and $\models_{\mathcal{M}} \psi' \rightarrow \psi[\sigma]$. We conclude $\models_{\mathcal{M}} \psi[\sigma]$.

$\neg$I : In this case $\psi = \neg\psi'$ and we have a derivation $\varphi_1, \ldots, \varphi_n, \psi' \vdash \bot$. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying $\{\varphi_1, \ldots, \varphi_n\}$. If $\mathcal{M}$ and σ also satisfy ψ' we conclude $\models_{\mathcal{M}} \bot[\sigma]$ from the induction hypothesis. The last statement is a contradiction so that we conclude $\not\models_{\mathcal{M}} \psi'[\sigma]$, and, hence, $\models_{\mathcal{M}} \psi[\sigma]$.

$\neg$E : In this case $\psi = \bot$ and we have derivations $\varphi_1, \ldots, \varphi_n \vdash \psi'$ and $\varphi_1, \ldots, \varphi_n \vdash \neg\psi'$ for some formula ψ'. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying the set of premises. From the induction hypothesis we get $\models_{\mathcal{M}} \psi'[\sigma]$ and $\models_{\mathcal{M}} \neg\psi'[\sigma]$. This is a contradiction so that such a pair $\mathcal{M}$ and σ does not exist showing $\varphi_1, \ldots, \varphi_n \models \bot$.

PBC : In this case we have a derivation $\varphi_1, \ldots, \varphi_n, \neg\psi \vdash \bot$. Now, assume $\mathcal{M}$ is a model and σ an environment satisfying $\{\varphi_1, \ldots, \varphi_n\}$. If $\mathcal{M}$ and σ do not satisfy ψ we conclude $\models_{\mathcal{M}} \bot[\sigma]$ from the induction hypothesis. The last statement is a contradiction so that we conclude $\models_{\mathcal{M}} \psi[\sigma]$.

$\forall$I : In this case $\psi = \forall x{:}\psi'$, and we have a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi'$. The variable condition implies that x does not occur free in any of $\varphi_1, \ldots, \varphi_n$. Assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. By Lemma 2.2.7(2) we have $\models_{\mathcal{M}} \varphi_i[\sigma[a/x]]$ for all $a \in |\mathcal{M}|$ and $i \in \{1, \ldots, n\}$. By the induction hypothesis we conclude $\models_{\mathcal{M}} \psi'[\sigma[a/x]]$ for all $a \in |\mathcal{M}|$, and, hence, $\models_{\mathcal{M}} \psi[\sigma]$.

$\forall$E : In this case $\psi = \psi'[t/x]$, and we have a derivation $\varphi_1, \ldots, \varphi_n \vdash \forall x{:}\psi'$. Assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. By the induction hypothesis we conclude $\models_{\mathcal{M}} \forall x{:}\psi'[\sigma]$, and, in particular, $\models_{\mathcal{M}} \psi'[\sigma[\bar{\sigma}(t)/x]]$. From Lemma 2.2.9(2) we get $\models_{\mathcal{M}} \psi'[t/x][\sigma]$.

$\exists$I : In this case $\psi = \exists x{:}\psi'$, and we have a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi'[t/x]$. Assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. By the induction hypothesis we conclude $\models_{\mathcal{M}} \psi'[t/x][\sigma]$. Lemma 2.2.9(2) implies $\models_{\mathcal{M}} \psi'[\sigma[\bar{\sigma}(t)/x]]$, and, hence, $\models_{\mathcal{M}} \psi[\sigma]$.

$\exists$E : In this case we have derivations $\varphi_1, \ldots, \varphi_n \vdash \exists x{:}\psi'$ and $\Sigma, \psi' \vdash \psi$ with $\Sigma \subseteq \{\varphi_1, \ldots, \varphi_n\}$. The variable condition implies that x does not occur free in ψ and in any formula of Σ. Assume $\mathcal{M}$ is a model and σ an environment with $\models_{\mathcal{M}} \varphi_i[\sigma]$ for $i \in \{1, \ldots, n\}$. By the induction hypothesis we get $\models_{\mathcal{M}} \exists x{:}\psi'[\sigma]$. We conclude that there is an $a \in |\mathcal{M}|$ with $\models_{\mathcal{M}} \psi'[\sigma[a/x]]$. Since x does not occur free in the formulas in Σ we get $\models_{\mathcal{M}} \varphi[\sigma[a/x]]$ for all $\varphi \in \Sigma$ by Lemma 2.2.9(2). From the induction hypothesis we derive $\models_{\mathcal{M}} \psi[\sigma[a/x]]$. Again by Lemma 2.2.9(2) we conclude $\models_{\mathcal{M}} \psi[\sigma]$ since x does not occur free in ψ. $\square$

First-order languages quite often include a predicate symbol for equality $=$. The main difference between the symbol $=$ and other predicate symbols is that $=$ usually has a predefined interpretation, i.e., $=^{\mathcal{M}}= \{(a, a) \mid a \in |\mathcal{M}|\}$ for all models. We refer to this variant as first-order logic with equality. An extended version of natural deduction provides an introduction and elimination rule for $=$:

$$\frac{}{t = t}\ =\text{I} \qquad \frac{t_1 = t_2 \quad \varphi[t_1/x]}{\varphi[t_2/x]}\ =\text{E}$$

Notice that $=$I does not have an assumption, i.e., this rule is actually an axiom.

Chapter 3

Formal Semantics of Programming Languages

In this chapter we will introduce a small imperative programming language. For this language we present its operational semantics, i.e., an abstract machine executing programs, as well as logical calculus to derive certain properties about programs.

3.1 IMP- An Imperative Programming Language

This chapter introduces the syntax of a small imperative programming language **IMP**. The language is based on constant, function, and predicate symbols for the natural numbers. In particular, we have the following sets of F^n of function symbols and P^n of predicate symbols of arity n:

- F^0 consists of all symbols for positive and negative integers with zero,
- $F^2 = \{+, -, *\}$,
- $P^0 = \{\texttt{true}, \texttt{false}\}$,
- $P^2 = \{\texttt{=}, \texttt{<=}\}$.

As before we will use infix notation and the standard precedence rules in the examples.

In addition, we will use Boolean expressions within programs. These expression are simply first-order formulas in our language that do not use quantifiers.

In order to define the syntax of **IMP**we will use a variant of grammar in BNF (Backus-Naur form). Within the rules we use certain symbols as meta-variables for syntactic entities, i.e., as so-called non-terminals. Concretely we use the following

- $x, y, \ldots$ are variables,
- a, a_0, a_1 range over arithmetic expressions, i.e., terms in the language above,
- b, b_0, b_1 range over boolean expressions, i.e., first-order formulas without quantifiers,
- c, c_0, c_1 range over commands (or programs).

The commands are defined by:

$$c ::= \texttt{skip} | x\texttt{:=}a | c_0; c_1 | \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1 \texttt{ fi} | \texttt{while } b \texttt{ do } c \texttt{ od}.$$

Throughout the rest of this chapter we will fix the integers with the standard interpretation for all function and predicate symbols as our model of interest. We denote this model simply by $\mathbb{Z}$. In particular, $\bar{\sigma}(a)$ is an integer for every arithmetic expression a and environment σ.

3.2 Operational Semantics of IMP

Operational semantics of a programming language consists of specifying an abstract machine executing programs. This corresponds to an implementation of the language.

The language **IMP** will modify the contents of variables. Consequently, a program c takes an environment σ, changes its content, and returns the modified environment σ'. This fact will be denoted by

$$\langle c, \sigma \rangle \to \sigma'.$$

$\rightarrow$ is called the evaluation relation. This relation is defined by the following rules:

$$\text{(Skip)} \qquad \langle \texttt{skip}, \sigma \rangle \rightarrow \sigma$$

$$\text{(Assignment)} \qquad \langle x := a, \sigma \rangle \rightarrow \sigma[\bar{\sigma}(a)/x]$$

$$\text{(Sequencing)} \qquad \frac{\langle c_0, \sigma \rangle \rightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \sigma'}$$

$$\text{(Conditional 1)} \qquad \frac{\langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1 \texttt{ fi}, \sigma \rangle \rightarrow \sigma'} \quad \text{iff } \models_{\mathbb{Z}} b[\sigma]$$

$$\text{(Conditional 2)} \qquad \frac{\langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1 \texttt{ fi}, \sigma \rangle \rightarrow \sigma'} \quad \text{iff } \models_{\mathbb{Z}} \neg b[\sigma]$$

$$\text{(Loop 1)} \qquad \langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma \rangle \rightarrow \sigma \quad \text{iff } \models_{\mathbb{Z}} \neg b[\sigma]$$

$$\text{(Loop 2)} \qquad \frac{\langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma'' \rangle \rightarrow \sigma'}{\langle \texttt{while } b \texttt{ do } c \texttt{ od}, \sigma \rangle \rightarrow \sigma'} \quad \text{iff } \models_{\mathbb{Z}} b[\sigma]$$

Several of the rules above have a side condition that has to be satisfied before the corresponding rule can be applied.

Notice that the evaluation relation is a partial function. Let w be the program `while true do skip od`, and σ be an arbitrary state. Then there is no state σ' with $\langle w, \sigma \rangle \rightarrow \sigma'$.

Before we continue we want to show that our programming language is deterministic, i.e., the relation on environments given by commands is, in fact, a partial function.

Lemma 3.2.1 *Let c be a command, and σ be a state. If $\langle c, \sigma \rangle \rightarrow \sigma_1$ and $\langle c, \sigma \rangle \rightarrow \sigma_2$, then $\sigma_1 = \sigma_2$ for all $\sigma_1, \sigma_2 \in \Sigma$.*

Proof. We prove the lemma by structural induction on the derivation of $\langle c, \sigma \rangle \rightarrow \sigma_1$.

(Skip) In this case c is `skip` and $\sigma = \sigma_1$. This implies that the derivation of $\langle c, \sigma \rangle \rightarrow \sigma_2$ also uses the rule (Skip), and, hence, $\sigma_1 = \sigma = \sigma_2$.

(Assignment) In this case c is $x := a$ for a location x and an expression a and $\sigma_1 = \sigma[\bar{\sigma}(a)/x]$. The second derivation must also uses this rule, and we get $\sigma_2 = \sigma[\bar{\sigma}(a)/x]$, i.e., $\sigma_1 = \sigma_2$.

(Sequencing) We have c is $c_0; c_1$ for commands c_0 and c_1. Furthermore, we have derivations $\langle c_0, \sigma\rangle \to \sigma_1'$ and $\langle c_1, \sigma_1'\rangle \to \sigma_1$ for a state σ_1'. The second derivation must use the same rule so that we get derivations $\langle c_0, \sigma\rangle \to \sigma_2'$ and $\langle c_1, \sigma_2'\rangle \to \sigma_2$ for a state σ_2'. By the induction hypothesis we first get $\sigma_1' = \sigma_2'$ and then $\sigma_1 = \sigma_2$.

(Conditional 1) In this case we have $\models_{\mathbb{Z}} b[\sigma]$ and $\langle c_0, \sigma\rangle \to \sigma_1$. Again, the second derivation must also use the rule (Conditional 1). We get a derivation $\langle c_0, \sigma\rangle \to \sigma_2$. The induction hypothesis implies $\sigma_1 = \sigma_2$.

(Conditional 2) Similar to the previous rule.

(Loop 1) We have $\models_{\mathbb{Z}} \neg b[\sigma]$ and $\sigma_1 = \sigma$. This implies that the second derivation uses (Loop 1) too, and we get $\sigma_1 = \sigma = \sigma_2$.

(Loop 2) We have $\models_{\mathbb{Z}} b[\sigma]$ and derivations $\langle c', \sigma\rangle \to \sigma_1'$ and $\langle c, \sigma_1'\rangle \to \sigma_1$. As in the previous cases the second derivation must also use (Loop 2) so that we obtain derivations $\langle c', \sigma\rangle \to \sigma_2'$ and $\langle c, \sigma_2'\rangle \to \sigma_2$. By the induction hypothesis we first get $\sigma_1' = \sigma_2'$ and then $\sigma_1 = \sigma_2$. □

The next lemma is concerned with the termination of a loop.

Lemma 3.2.2 *If* $\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}, \sigma\rangle \to \sigma'$, *then there is an* $n \in \mathbb{N}$ *and* σ_i *for* $i \in \{0, \ldots, n\}$ *with*

1. $\sigma_0 = \sigma$,
2. $\models_{\mathbb{Z}} b[\sigma_i]$ *and* $\langle c, \sigma_i\rangle \to \sigma_{i+1}$ *for* $i \in \{0, \ldots, n-1\}$,
3. $\sigma_n = \sigma'$ *and* $\models_{\mathbb{Z}} \neg b[\sigma_n]$.

Proof. Let n be the number of applications of the (Loop 2) rule with a conclusion that contains the program `while` b `do` c `od`. We show by mathematical induction (for n) that there are environments as required by 1.-3. above. If $n = 0$, then the derivation must be an application of the (Loop 1) rule. The environments $\sigma_0 = \sigma$ satisfy the assertion. Now, assume we have $n + 1$ applications of (Loop 2) of the form required. Then the last rule must be an application of (Loop 2). This implies $\models_{\mathbb{Z}} b[\sigma]$, and we get two derivations $\langle c, \sigma\rangle \to \sigma''$ and $\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{od}, \sigma''\rangle \to \sigma'$. The first derivation does not contain another (Loop 2) of the required form so that the second derivation has n occurrences. By the induction hypothesis we obtain environments with

1. $\sigma_0 = \sigma''$,

2. $\models_{\mathbb{Z}} b[\sigma_i]$ and $\langle c, \sigma_i \rangle \to \sigma_{i+1}$ for $i \in \{0, \ldots, n-1\}$,

3. $\sigma_n = \sigma'$ and $\models_{\mathbb{Z}} \neg b[\sigma_n]$.

These together with σ satisfy the assertion. □

3.3 Axiomatic Semantics of IMP

An axiomatic semantics for a programming language consists of a logic for deriving properties of those programs. In our case we are interested in partial correctness assertions. A partial correctness assertion for a command has the form:

$$\{\varphi\} c \{\psi\}$$

where φ, ψ are formulas and c is a command. Notice that the partial correctness assertion is not a first-order formula. φ is called the precondition and ψ the post-condition of the assertion. Informally a partial correctness assertion says that if the precondition is satisfied and the program terminates, then the postcondition is satisfied. Since the assertion does not make any statements about termination of the command it is called a partial correctness assertion. Formally, we define the satisfaction relation $\models \{\varphi\} c \{\psi\}[\sigma]$ with respect to an environment σ, i.e., whether the triple is true or not, by the following property:

$$\text{If } \models_{\mathbb{Z}} \varphi[\sigma] \text{ and } \langle c, \sigma \rangle \to \sigma', \text{ then } \models_{\mathbb{Z}} \psi[\sigma'].$$

As before we write $\models \{\varphi\} c \{\psi\}$ if $\models \{\varphi\} c \{\psi\}[\sigma]$ holds for all environment σ.

Now, we are going to present proof rules for partial correctness assertions. Those

rules are often called Hoare rules, and the proof system Hoare logic.

$$\text{(Skip)} \qquad \{\varphi\}\texttt{skip}\{\varphi\}$$

$$\text{(Assignment)} \qquad \{\psi[a/x]\}x := a\{\psi\}$$

$$\text{(Sequencing)} \qquad \frac{\{\varphi\}c_0\{\chi\} \quad \{\chi\}c_1\{\psi\}}{\{\varphi\}c_0; c_1\{\psi\}}$$

$$\text{(Conditional)} \qquad \frac{\{\varphi \wedge b\}c_0\{\psi\} \quad \{\varphi \wedge \neg b\}c_1\{\psi\}}{\{\varphi\}\texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1 \texttt{ fi}\{\psi\}}$$

$$\text{(Loop)} \qquad \frac{\{\varphi \wedge b\}c\{\varphi\}}{\{\varphi\}\texttt{while } b \texttt{ do } c \texttt{ od}\{\varphi \wedge \neg b\}}$$

$$\text{(Consequence)} \qquad \frac{\models \varphi \to \varphi' \quad \{\varphi'\}c\{\psi'\} \quad \models \psi' \to \psi}{\{\varphi\}c\{\psi\}}$$

Notice that the consequence rule relies on validity and not on the provability of two implications. A partial correction assertion $\{\varphi\}c\{\psi\}$ derived using the rules above is called a theorem, and we denote that fact by $\vdash \{\varphi\}c\{\psi\}$.

3.3.1 Soundness

It is not yet clear that theorems of the Hoare logic are valid actually valid partial correctness assertions. The corresponding property is called soundness of the logic, which we are going to prove in this section. Before we are able to do this we need a lemma relating substitution and validity.

Now we are ready to establish the soundness of the proof rules.

Theorem 3.3.1 (Soundness) *Let $\{\varphi\}c\{\psi\}$ be a partial correctness assertion. Then $\vdash \{\varphi\}c\{\psi\}$ implies $\models \{\varphi\}c\{\psi\}$.*

Proof. The proof uses structural induction on the derivation $\vdash \{\varphi\}c\{\psi\}$.

(Skip): In this case $\psi = \varphi$ and $c = \texttt{skip}$. Assume σ is an environment with $\models_{\mathbb{Z}} \varphi[\sigma]$. Since $\langle c, \sigma\rangle \to \sigma$ we get $\models_{\mathbb{Z}} \psi[\sigma]$.

(Assignment): Assume $\models_{\mathbb{Z}} \psi[a/x][\sigma]$. By Lemma 2.2.9 this is equivalent to the statement $\models_{\mathbb{Z}} \psi[\sigma[\bar{\sigma}(a)/x]]$. Since $\langle x := a, \sigma\rangle \to \sigma[\bar{\sigma}(a)/x]$ we conclude the soundness of the assignment rule.

(Sequencing): In this case we have two derivations $\vdash \{\varphi\}c_0\{\chi\}$ (left subtree) and $\vdash \{\chi\}c_1\{\psi\}$ (right subtree). Now, assume $\models_{\mathbb{Z}} \varphi[\sigma]$ and $\langle c_0; c_1, \sigma\rangle \to \sigma'$. From the operational semantics we conclude that $\langle c_0, \sigma\rangle \to \sigma''$ and $\langle c_1, \sigma''\rangle \to \sigma'$ for some environment σ''. The induction hypothesis for the left subtree shows $\models_{\mathbb{Z}} \chi[\sigma'']$. Using the induction hypothesis for the right subtree we get $\models_{\mathbb{Z}} \psi[\sigma']$.

(Conditional): In this case $c = \texttt{if}\ b\ \texttt{then}\ c_0\ \texttt{else}\ c_1\ \texttt{fi}$ and we have two derivations $\vdash \{\varphi \wedge b\}c_0\{\psi\}$ (left subtree) and $\vdash \{\varphi \wedge \neg b\}c_1\{\psi\}$ (right subtree). Now, assume $\models_{\mathbb{Z}} \varphi[\sigma]$ and $\langle c, \sigma\rangle \to \sigma'$. If $\models_{\mathbb{Z}} b[\sigma]$, then we have $\models_{\mathbb{Z}} \varphi \wedge b[\sigma]$. Furthermore, from the operational semantics we obtain $\langle c_0, \sigma\rangle \to \sigma'$. Using the induction hypothesis for the left subtree we get $\models_{\mathbb{Z}} \psi[\sigma']$. The case $\not\models_{\mathbb{Z}} b[\sigma]$, i.e., $\models_{\mathbb{Z}} \neg b[\sigma]$ follows analogously using the induction hypothesis for the right subtree.

(Loop): In this case we have $c = \texttt{while}\ b\ \texttt{do}\ c'\ \texttt{od}$ and $\psi = \varphi \wedge \neg b$. Furthermore, we have a subtree $\vdash \{\varphi \wedge b\}c'\{\varphi\}$. Now, assume $\models_{\mathbb{Z}} \varphi[\sigma]$ and $\langle c, \sigma\rangle \to \sigma'$. By Lemma 3.2.2 there is an $n \in \mathbb{N}$ and σ_i for $i \in \{0, \ldots, n\}$ with

1. $\sigma_0 = \sigma$,
2. $\models_{\mathbb{Z}} b[\sigma_i]$ and $\langle c', \sigma_i\rangle \to \sigma_{i+1}$ for $i \in \{0, \ldots, n-1\}$,
3. $\sigma_n = \sigma'$ and $\models_{\mathbb{Z}} \neg b[\sigma_n]$.

We show by mathematical induction that $\models_{\mathbb{Z}} \varphi[\sigma_i]$ for all $i \in \{0, \ldots, n\}$. If $i = 0$, then $\sigma_i = \sigma$ so that the assertion follows from the general assumption in this case. For the induction step we have $\models_{\mathbb{Z}} \varphi[\sigma_i]$ by the induction hypothesis for i so that $\models_{\mathbb{Z}} \varphi \wedge b[\sigma_i]$ follows from 2. since $i + 1 \leq n$. Together with $\langle c', \sigma_i\rangle \to \sigma_{i+1}$ and the induction hypothesis for the subtree $\vdash \{\varphi \wedge b\}c'\{\varphi\}$ we obtain $\models_{\mathbb{Z}} \varphi[\sigma_{i+1}]$. This completes the induction on i.

We conclude $\models_{\mathbb{Z}} \varphi[\sigma']$ and $\models_{\mathbb{Z}} \neg b[\sigma']$ from 3. and the induction above. This shows $\models_{\mathbb{Z}} \psi[\sigma']$.

(Consequence): Assume $\models_{\mathbb{Z}} \varphi[\sigma]$ and $\langle c, \sigma\rangle \to \sigma'$. From $\models_{\mathbb{Z}} \varphi \to \varphi'$ we conclude $\models_{\mathbb{Z}} \varphi \to \varphi'[\sigma]$, and, hence, $\models_{\mathbb{Z}} \varphi'[\sigma]$. By the induction hypothesis we obtain $\models_{\mathbb{Z}} \psi'[\sigma']$. Using $\models_{\mathbb{Z}} \psi' \to \psi$ we obtain $\models_{\mathbb{Z}} \psi[\sigma']$. □

The converse implication of the previous theorem is called completeness of the logic. It says that every valid partial correctness assertion can be derived using Hoare

logic. In other words the set of rules is sufficient for proving partial correctness of programs. The proof of this theorem is out of the scope of this course.

Chapter 4

Algebraic Specifications

An algebraic specification is specification, i.e. a formal description, of (several) abstract data types and their operations. In this approach all operations are functions and their behavior is described by formulas.

4.1 Syntax: Signatures

First we want to introduce the notion of a signature. A signature contains the syntactic material necessary for an algebraic specification.

Definition 4.1.1 *A signature* $\mathrm{SIG} = (S, C, F)$ *is a triple of three pairwise disjoint sets:*

- S*, the set of sorts,*
- C*, the set of constant symbols, is the union of pairwise disjoint sets* C_s *of constant symbols of sort* $s \in S$*,*
- F*, the set of function symbols, is the union of pairwise disjoint sets* $F_{s_1,\ldots,s_n;s}$ *of function symbols with argument sorts* $s_1, \ldots, s_n \in S$ *and range sort* $s \in S$*.*

Similar to set Term of terms in first order logic we define expressions or terms of a signature. Hereby we have to take care that terms are well-typed, i.e. that function symbols are just applied to terms of correct sort. As before we want to be able to use variables to construct terms. Therefore, let X be a union of pairwise disjoint sets X_s of variables of sort $s \in S$, called a collection of variables.

Definition 4.1.2 *Let* SIG *be a signature, and X be a variable collection. Then the set $T_s^{\mathrm{SIG}}(X)$ of terms of sort $s \in S$ is defined by:*

1. *$X_s \subseteq T_s^{\mathrm{SIG}}(X)$, i.e. every variable of sort s is a term of sort s,*
2. *$C_s \subseteq T_s^{\mathrm{SIG}}(X)$, i.e. every constant of sort s is a term of sort s,*
3. *If $t_1 \in T_{s_1}^{\mathrm{SIG}}(X), \ldots, t_n \in T_{s_n}^{\mathrm{SIG}}(X)$ and $f \in F_{s_1,\ldots,s_n;s}$ then $f(t_1, \ldots, t_n) \in T_s^{\mathrm{SIG}}(X)$.*

The definition of the set of free variables $FV(t)$ of a term t is obvious. t is called a ground term if $FV(t) = \emptyset$, i.e. t does not contain any variable. Similar to the set Term we define the substitution $t[t'/x]$ of a term $t' \in T_s^{\mathrm{SIG}}(X)$ for a variable $x \in X_s$ in the term t. Notice that t' and x must be of the same sort in order to guarantee that $t[t'/x]$ is well-typed. The details of the definition are left to the reader.

In order to specify the behavior of the functions of a signature we need a notion of formulas. Formulas are constructed similar to the set FOL of first-order formulas. As we did for terms above we have to make sure that all expressions are well-typed.

Definition 4.1.3 *Let* SIG *be a signature, and X be a variable collection. Then the set $F^{\mathrm{SIG}}(X)$ of formulas is defined by:*

1. *If $t_1, t_2 \in T_s^{\mathrm{SIG}}(X)$, i.e. terms of same sort, then $t_1 = t_2 \in F^{\mathrm{SIG}}(X)$.*
2. *If $\varphi \in F^{\mathrm{SIG}}(X)$, then $\neg\varphi \in F^{\mathrm{SIG}}(X)$.*
3. *If $\varphi_1, \varphi_2 \in F^{\mathrm{SIG}}(X)$, then $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \to \varphi_2 \in F^{\mathrm{SIG}}(X)$.*
4. *If $\varphi \in F^{\mathrm{SIG}}(X)$ and $x \in X_s$, then $\forall x \in s : \varphi, \exists x \in s : \varphi \in F^{\mathrm{SIG}}(X)$.*

The definition of the free variables of a formula is again obvious. Similar to first-order logic we define the substitution $\varphi[t/x]$ of a term $t \in T_s^{\mathrm{SIG}}(X)$ for a variable $x \in X_s$ in the formula φ. Again t' and x must be of the same sort in order to guarantee that $\varphi[t'/x]$ is well-formed. As above we leave the details of this definition to the reader.

Definition 4.1.4 *An algebraic specification* $\mathrm{SPEC} = (\mathrm{SIG}, X, E)$ *consists of a signature* SIG*, a collection of variables X and set of formulas $E \subseteq F^{\mathrm{SIG}}(X)$.*

4.2 Semantics: SIG-Algebras and Models

We start with the definition of a SIG-algebra providing an interpretation of all syntactic entities of a signature. We are going to interpret function symbols by partial functions. The reason is that a program implementing a an operation might not terminate for certain input, i.e. is in fact a partial function.

Definition 4.2.1 *Let* $\mathrm{SIG} = (S, C, F)$ *be a signature. A triple* $A = (S^A, C^A, F^A)$ *is called a* SIG*-algebra iff:*

- $S^A = \{s^A \mid s \in S\}$ *is a set of non-empty sets* s^A *for each sort* $s \in S$*, called the base sets or domains of* A*.*
- $C^A = \{c^A \mid c \in C\}$ *is a set of elements* c^A *so that* $c^A \in s^A$ *iff* $c \in C_s$*.*
- $F^A = \{f^A \mid f \in F\}$ *is a set of partial functions* f^A *so that* $f^A : s_1^A \times \cdots \times s_n^A \to s^A$*, i.e.* f^A *is a partial function taking elements from* $s_1^A, \ldots, s_n^A$ *as arguments returning a value from* s^A*, iff* $f \in F_{s_1,\ldots,s_n;s}$*.*

In order to compute the value of a term or the validity of a formula we need elements for each variable (see state and interpretation). An assignment $a : X \to \bigcup_{s \in S} s^A$ is a function mapping variable to elements in the domains of A so that $a(x) \in S^A$ iff $x \in X_s$. If $x \in X_s$ and $u \in s^A$ we denote by $a[u/x]$ the assignment obtained from a by changing the content for x to u.

Definition 4.2.2 *Let* $\mathrm{SIG} = (S, C, F)$ *be a signature,* X *be a collection of variables,* A *be a* SIG*-algebra, and* a *be an assignment. Then the value* $\mathcal{V}_A(t)a \in s^A$ *of a term* $t \in T_s^{\mathrm{SIG}}(X)$ *in the* SIG*-algebra* A *and the assignment* a *is defined by:*

- $\mathcal{V}_A(x)a = a(x)$.
- $\mathcal{V}_A(c)a = c^A$.
- $\mathcal{V}_A((f(t_1, \ldots, t_n)))a = \begin{cases} f^A(\mathcal{V}_A(t_1)a, \ldots, \mathcal{V}_A(t_n)a) & \text{if } \mathcal{V}_A(t_i)a \text{ for } 1 \le i \le n \\ & \text{and } f^A(\mathcal{V}_A(t_1)a, \ldots, \mathcal{V}_A(t_n)a) \\ & \text{are defined} \\ \text{undefined} & \text{otherwise} \end{cases}$

Notice that the value function is a partial function itself.

In the following a statement $\mathcal{V}_A(t_1)a = \mathcal{V}_A(t_2)a$ always reads as follows: Both values $\mathcal{V}_A(t_1)a$ and $\mathcal{V}_A(t_2)a$ are defined and they are equal or both sides are undefined.

The value of ground term t does not depend on the assignment since t contains no variable. We omit a formal proof of this fact, and we write $\mathcal{V}_A(t)$ instead $\mathcal{V}_A(t)a$ with an arbitrary assignment a.

A SIG-algebra may contain junk, i.e. elements that are not described by any ground term of the signature. In computer science we are usually not interested in those algebras. Main reason is that it is just possible to implement a data type with at most countable many elements. Therefore, we call a SIG-algebra term generated (or finitely generated) if for every sort $s \in S$ and every $u \in s^A$ there is a term $t \in T^{\mathrm{SIG}}(X)$ with $\mathcal{V}_A(t) = u$.

Definition 4.2.3 *Let* $\mathrm{SIG} = (S, C, F)$ *be a signature,* X *be a collection of variables,* A *be a* SIG*-algebra, and* a *be an assignment. Then the satisfaction relation* $(A, a) \models \varphi$ *of a formula* $\varphi \in F^{\mathrm{SIG}}(X)$ *in the* SIG*-algebra* A *and the assignment* a *is defined by:*

- $(A, a) \models t_1 = t_2$ *if* $\mathcal{V}_A(t_1)a = \mathcal{V}_A(t_2)a$.
- $(A, a) \models \neg\varphi$ *if not* $(A, a) \models \varphi$.
- $(A, a) \models \varphi_0 \wedge \varphi_1$ *if* $(A, a) \models \varphi_0$ *and* $(A, a) \models \varphi_1$.
- $(A, a) \models \varphi_0 \vee \varphi_1$ *if* $(A, a) \models \varphi_0$ *or* $(A, a) \models \varphi_1$.
- $(A, a) \models \varphi_0 \to \varphi_1$ *if* $(A, a) \models \varphi_0$ *implies* $(A, a) \models \varphi_1$.
- $(A, a) \models \forall x \in s : \varphi$ *if for all* $u \in s^A$ *we have* $(A, a[u/x]) \models \varphi$.
- $(A, a) \models \forall x \in s : \varphi$ *if there is a* $u \in s^A$ *with* $(A, a[u/x]) \models \varphi$.

We write $A \models \varphi$ *iff* $(A, a) \models \varphi$ *for all assignments* a *and* $\models \varphi$ *iff* $A \models \varphi$ *for all* SIG*-algebras* A.

The definition above is very similar to the definition of the semantics of first-order logic. The only remarkable difference is in the reading of $=$ in the presence of partial functions.

Definition 4.2.4 *Let* $\mathrm{SPEC} = (\mathrm{SIG}, X, E)$ *be an algebraic specification. Then:*

1. *A* SIG*-algebra* A *is called model of* SPEC *if* $A \models \varphi$ *for all* $\varphi \in E$.

2. *The semantics of* SPEC *is the class* Gen(SPEC) *of all term generated models of* SPEC.

The approach taken here is called loose semantics because we consider all possible term generated models. Other approaches define the semantics of an algebraic specification by choosing a particular model, e.g. the initial or the terminal model. We will study those models in the next section.

4.3 Homomorphisms, initial and terminal Models

Functions between structures of the same kind preserving the structure are usually called homomorphisms. In terms of SIG-algebras we get the following definition.

Definition 4.3.1 *Let* SIG *be a signature, and* A *and* B *be* SIG*-algebras. A family* $H = (h_s)_{s \in S}$ *of partial function* $h_s : s^A \to s^B$ *is called a* SIG*-homomorphism from* A *to* B *if*

1. $h_s(c^A) = c^B$ *for all* $c \in C_s$,

2. $h_s(f^A(u_1, \ldots, u_n)) = f^B(h_{s_1}(u_1), \ldots, h_{s_n}(u_n))$ *for all* $f \in F_{s_1,\ldots,s_n;s}$ *and* $u_1 \in s_1^A, \ldots, u_n \in s_n^A$.

Notice that if the right hand side of the equation in 2. is not defined, then so must the left hand side. We may visualize the second property by the following diagram where we assume for simplicity that f is a unary function symbol:

$$\begin{array}{ccc} s^A & \xrightarrow{f^A} & s'^A \\ \downarrow{\scriptstyle h_s} & & \downarrow{\scriptstyle h_{s'}} \\ s^B & \xrightarrow{f^B} & s'^B \end{array}$$

If H is a SIG-homomorphism from A to B we write $H : A \to B$.

Lemma 4.3.2 *Let* SIG *be a signature,* A, B, C *and* D *be* SIG*-algebras, and* $H : A \to B$, $K : B \to C$ *and* $L : C \to D$ *be* SIG*-homomorphisms. Then we have*

1. $\mathrm{Id}_A := (\mathrm{id}_{s^A})_{s \in S}$, *i.e. the family of the identity function on each domain, is a* SIG*-homomorphism from* A *to* A,

2. $K \circ H := (k_s \circ h_s)_{s \in S}$ *is a* SIG*-homomorphism from* A *to* C*,*

3. $(L \circ K) \circ H = L \circ (K \circ H)$*, i.e. composition of* SIG*-homomorphisms is associative,*

4. $\mathrm{Id}_B \circ H = H = H \circ \mathrm{Id}_A$*, i.e. the identity homomorphism is a left and right neutral element for composition,*

5. *If every* h_s *is bijective, then* $H^{-1} := (h_s^{-1})_{s \in S}$ *is a* SIG*-homomorphism from* B *to* A *and* $H \circ H^{-1} = \mathrm{Id}_B$ *and* $H^{-1} \circ H = \mathrm{Id}_A$*.*

Proof.

1. This is trivial since both sides of both equations in Definition 4.3.1 become equal.

2. The assertion follows immediately from

$$
\begin{aligned}
&(k_s \circ h_s)(c^A) \\
&= k_s(h_s(c^A)) \\
&= k_s(c^B) && \text{since } H \text{ is a homomorphism} \\
&= c^C, && \text{since } K \text{ is a homomorphism} \\
&(k_s \circ h_s)(f^A(u_1, \ldots, u_n)) \\
&= k_s(h_s(f^A(u_1, \ldots, u_n))) \\
&= k_s(f^B(h_{s_1}(u_1), \ldots, h_{s_n}(u_n))) && \text{since } H \text{ is a homomorphism} \\
&= f^C(k_{s_1}(h_{s_1}(u_1)), \ldots, k_{s_n}(h_{s_n}(u_n))) && \text{since } K \text{ is a homomorphism} \\
&= f^C((k_{s_1} \circ h_{s_1})(u_1), \ldots, (k_{s_n} \circ h_{s_n})(u_n)).
\end{aligned}
$$

3. This follows immediately from the fact that composition of function is associative.

4. This follows from the fact that the identity function on s^A (resp. on s^B) is a right (resp. left) neutral element for composition.

5. Consider the computations

$$
\begin{aligned}
& h_s^{-1}(c^B) \\
& = h_s^{-1}(h_s(c^A)) && \text{since } H \text{ is a homomorphism} \\
& = c^A, \\
& h_s^{-1}(f^B(v_1, \ldots, v_n)) \\
& = h_s^{-1}(f^B(h_{s_1}(u_1), \ldots, h_{s_n}(u_n))) && \text{for some } u_1, \ldots, u_n \in s^A \\
& && \text{since each } h_{s_i} \text{ is surjective} \\
& = h_s^{-1}(h_s(f^A(u_1, \ldots, u_n))) && \text{since } H \text{ is a homomorphism} \\
& = f^A(u_1, \ldots, u_n).
\end{aligned}
$$

This shows that H^{-1} is a SIG-homomorphism from B to A. The remaining properties follow immediately from the corresponding properties for each h_s and h_s^{-1}. □

Notice that 1.-4. of the previous lemma shows that the structure of SIG-algebras with SIG-homomorphisms is a category. Categories play an important rule theoretical computer science and mathematics.

A SIG-homomorphism H so that H^{-1} exists is called a SIG-isomorphism. Two SIG-algebras are called isomorphic (denoted by $A \cong B$) if there is a SIG-isomorphism $H : A \to B$ (or equivalently $K : B \to A$).

Lemma 4.3.3 *Let* SIG *be a signature, A and B be* SIG*-algebras, $H : A \to B$ be a* SIG*-homomorphism, and t a ground term. If $\mathcal{V}_B(t)$ or $h_s(\mathcal{V}_A(t))$ are defined, then so is the other value and we have $h_s(\mathcal{V}_A(t)) = \mathcal{V}_B(t)$.*

Proof. This is shown by structural induction on t.

$t \equiv c$**:** In this case we have $\mathcal{V}_B(()t) = c^B$ and $\mathcal{V}_A(()t) = c^A$ so that both values are defined. From the definition of a SIG-homomorphism we conclude $h_s(\mathcal{V}_A(t)) = h_s(c^A) = c^B = \mathcal{V}_B(t)$ and, in particular, that $h_s(\mathcal{V}_A(t))$ is defined.

$t \equiv f(t_1, \ldots, t_n)$**:** If $\mathcal{V}_B(t)$ is defined, we have $\mathcal{V}_B(t) = f^B(\mathcal{V}_B(t_1), \ldots, \mathcal{V}_B(t_n))$. From the induction hypothesis we conclude $h_{s_i}(\mathcal{V}_A(t_i))$ are defined and $\mathcal{V}_B(t_i) = h_{s_i}(\mathcal{V}_A(t_i))$ for $1 \leq i \leq n$ since the value of each subterm t_i of t is defined in B. We obtain $\mathcal{V}_B(t) = f^B(h_{s_1}(\mathcal{V}_A(t_i)), \ldots, h_{s_n}(\mathcal{V}_A(t_i)))$. From the definition

of a SIG-homomorphism we derive that $h_s(f^A(\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n))$ is defined and equal to $\mathcal{V}_B(t)$. We conclude

$$\mathcal{V}_B(t) = h_s(f^A(\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n)) = h_s(\mathcal{V}_A(t)).$$

The latter also shows that $h_s(\mathcal{V}_A(t))$ is defined.

If $h_s(\mathcal{V}_A(t))$ is defined, then so is $\mathcal{V}_A(t)$ and we have

$$h_s(\mathcal{V}_A(t)) = h_s(f^A(\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n)).$$

From the definition of a SIG-homomorphism we conclude

$$h_s(\mathcal{V}_A(t)) = f^B(h_{s_1}(\mathcal{V}_A(t_1)), \ldots, h_{s_n}(\mathcal{V}_A(t_n))).$$

In particular, all $h_{s_i}(\mathcal{V}_A(t_1))$ with $1 \leq i \leq n$ are defined. From the induction hypothesis we conclude that $\mathcal{V}_B(t_i)$ is defined and that $h_{s_i}(\mathcal{V}_A(t_1)) = \mathcal{V}_B(t_i)$ for all $1 \leq i \leq n$. This implies $h_s(\mathcal{V}_A(t)) = f^B(\mathcal{V}_B(t_1), \ldots, \mathcal{V}_B(t_n)) = \mathcal{V}_B(t)$. The latter also shows that $\mathcal{V}_B(t)$ is defined. □

We get the following lemma as an immediate consequence from the previous one.

Lemma 4.3.4 *Let* SIG *be a signature, A a term generated* SIG*-algebra, and B be an arbitrary* SIG*-algebra. Then there is at most one* SIG*-homomorphism from A to B.*

Proof. Assume $H, K : A \to B$ are SIG-homomorphisms. We show that $h_s(\mathcal{V}_A(t)) = k_s(\mathcal{V}_A(t))$ for all ground terms t. The assertion then follows from the fact that A is term generated. Notice that the equality stated above is an equality of partially defined values. We have to show that if $h_s(\mathcal{V}_A(t))$ is defined, then $k_s(\mathcal{V}_A(t))$ is defined and $h_s(\mathcal{V}_A(t)) = k_s(\mathcal{V}_A(t))$. Assume $h_s(\mathcal{V}_A(t))$ is defined. From Lemma 4.3.3 we obtain that $\mathcal{V}_B(t)$ is defined and that $h_s(\mathcal{V}_A(t)) = \mathcal{V}_B(t)$. Applying Lemma 4.3.3 again we derive that $k_s(\mathcal{V}_A(t))$ is defined and that $k_s(\mathcal{V}_A(t)) = \mathcal{V}_B(t)$. We conclude $h_s(\mathcal{V}_A(t)) = k_s(\mathcal{V}_A(t))$. □

One may define a pre-order on SIG-algebras by $A \preceq B$ iff there is a homomorphism from B to A. From the previous lemma it follows that if $A \preceq B$ and $B \preceq A$, then $A \cong B$ (see also Theorem 4.3.6). Among the models of a specification we want to identify the least and the greatest element (if they exist).

Definition 4.3.5 *Let* SPEC *be an algebraic specification. Then we call*

- *a (term generated) model I from* Gen(SPEC) *initial if there is a* SIG*-homomorphism from I to any model in* Gen(SPEC)*.*
- *a (term generated) model T from* Gen(SPEC) *terminal if there is a* SIG*-homomorphism from any model in* Gen(SPEC) *to T.*

The notions of initial and terminal objects are general notions in category theory. Usually it requires for initial objects I that there is exactly one morphism to every other object of the category. In our case we are able to drop the uniqueness property since we have already shown that there at most one SIG-homomorphism between term generated SIG-algebras. However, initial and terminal objects are unique up to isomorphism.

Theorem 4.3.6 *Let* SPEC *be an algebraic specification. Then we have:*

1. *An initial model is unique up to isomorphism, i.e. if I_1 and I_2 are initial, then $I_1 \cong I_2$.*
2. *A terminal model is unique up to isomorphism, i.e. if I_1 and I_2 are initial, then $I_1 \cong I_2$.*

Proof.

1. Assume I_1 and I_2 are initial models. Then there is a SIG-homomorphism $H : I_1 \to I_2$ since I_1 is initial and a SIG-homomorphism $K : I_2 \to I_1$ since I_2 is initial. Then $K \circ H$ is a SIG-homomorphism from I_1 to I_1 by Lemma 4.3.2(2). The same lemma also shows that Id_{I_1} is a SIG-homomorphism from I_1 to I_1. From Lemma 4.3.4 we conclude $K \circ H = \mathrm{Id}_{I_1}$. Analogously we obtain $H \circ K = \mathrm{Id}_{I_2}$. This shows that we have $h_s = k_s^{-1}$ for every sort $s \in S$, i.e. that H (and K) is an isomorphism.
2. Analogously to 1. □

In the presence of partial operations, i.e. function symbols are interpreted by partial functions, a terminal model does usually not exist. For this reason we want to concentrate on initial models.

In the remainder of this section we want to find sufficient properties of the algebraic specification so that initial model exists. Those properties are essentially properties of the set E of axioms. First we consider the case of $E = \emptyset$.

Definition 4.3.7 *Let* SIG $= (S, C, F)$ *be a signature. Then we define the term algebra* Term *by*

- s^{Term} *is the set of ground terms of sort* s*,*
- $c^{\text{Term}} = c$*, i.e. the interpretation of a constant symbol is the symbol itself,*
- $f^{\text{Term}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ *for all* $f \in F_{s_1,\ldots,s_n;s}$ *and* $t_1 \in s_1^{\text{Term}}, \ldots, t_n \in s_n^{\text{Term}}$*.*

Theorem 4.3.8 *Let* SIG *be a signature. Then* Term *the initial model of* $(\text{SIG}, \emptyset)$*.*

Proof. Let A be a term generated SIG-algebra. Then we define a family of functions $H : \text{Term} \to A$ by

$$h_s(t) := \mathcal{V}_A(t).$$

Let $c \in C_s$ be a constant symbol. Then we have $h_s(c^{\text{Term}}) = h_s(c) = \mathcal{V}_A(c) = c^A$. Now assume $f \in F_{s_1,\ldots,s_n;s}$ and $t_1 \in s_1^{\text{Term}}, \ldots, t_n \in s_n^{\text{Term}}$. Then

$$\begin{aligned} h_s(f^{\text{Term}}(t_1, \ldots, t_n)) &= h_s(f(t_1, \ldots, t_n)) \\ &= \mathcal{V}_A(f(t_1, \ldots, t_n)) \\ &= f^A(\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n)) \\ &= f^A(h_{s_1}(t_1), \ldots, h_{s_n}(t_n)). \end{aligned}$$

This completes the proof. □

Notice that the above theorem is not restricted to the class of term generated SIG-algebras. For the more general version we have to show that the SIG-homomorphism defined above is unique. But this follows immediately since Term is term generated.

With no axioms available the term model is the initial model of the specification.

In order to model the effect of certain axioms on the term model we need the notion of a congruence.

Definition 4.3.9 *Let* SIG *be a signature, and* A *be a* SIG*-algebra. We call a family* $\approx = (\sim_s)_{s \in S}$ *of equivalence relations* $\sim_s$ *on* s^A*, i.e.* $\sim_s \subseteq s^A \times s^A$*, a* SIG*-congruence on* A *if*

1. $u_1 \sim_{s_1} v_1, \ldots, u_n \sim_{s_n} v_n$ *and* $f^A(u_1, \ldots, u_n)$ *is defined implies that* $f^A(v_1, \ldots, v_n)$ *is defined and* $f^A(u_1, \ldots, u_n) \sim_s f^A(v_1, \ldots, v_n)$*,*

2. $u_1 \sim_{s_1} v_1, \ldots, u_n \sim_{s_n} v_n$ *and* $f^A(v_1, \ldots, v_n)$ *is defined implies that* $f^A(u_1, \ldots, u_n)$ *is defined and* $f^A(u_1, \ldots, u_n) \sim_s f^A(v_1, \ldots, v_n)$,

for all $f \in F_{s_1,\ldots,s_n;s}$.

Given a SIG-congruence we are able to construct a new SIG-algebra based on the equivalence classes of the congruence.

Definition 4.3.10 *Let* SIG *be a signature,* A *be a* SIG*-algebra, and* $\approx$ *be a* SIG*-congruence on* A. *Then the quotient algebra* $A/\approx$ *is defined by*

- $s^{A/\approx} = \{[u]_{\sim_s} \mid u \in s^A\}$, *i.e. the elements of* $s^{A/\approx}$ *are the equivalence classes of elements of* s^A *with respect to* $\sim_s$,
- $c^{A/\approx} = [c^A]$,
- $f^{A/\approx}([u_1], \ldots, [u_n]) = [f^A(u_1, \ldots, u_n)]$.

Notice that $f^{A/\approx}$ is well-defined because $\approx$ is a SIG-congruence.

Theorem 4.3.11 *Let* SPEC $= (\mathrm{SIG}, E)$ *be an algebraic specification. Then the family of relation* $\approx = (\sim_s)_{s \in S}$ *on the term model* Term *defined by*

$$t_1 \sim_s t_2 \text{ iff } \mathcal{V}_A(t_1) = \mathcal{V}_A(t_2) \text{ for all } A \in \mathrm{Gen(SPEC)}$$

is a SIG*-congruence on* Term.

Proof. Assume $t_1 \sim_{s_1} t'_1, \ldots, t_n \sim_{s_n} t'_n$, i.e. $\mathcal{V}_A(t_1) = \mathcal{V}_A(t'_1), \ldots, \mathcal{V}_A(t_n) = \mathcal{V}_A(t'_n)$ for all $A \in \mathrm{Gen(SPEC)}$. Since both $f^{\mathrm{Term}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ and $f^{\mathrm{Term}}(t'_1, \ldots, t'_n) = f(t'_1, \ldots, t'_n)$ are defined it remains to show that $f(t_1, \ldots, t_n) \sim_s f(t'_1, \ldots, t'_n)$. Therefore, let $A \in \mathrm{Gen(SPEC)}$. Suppose $\mathcal{V}_A(f(t_1, \ldots, t_n))$ is not defined. Then either $\mathcal{V}_A(t_i)$ for an $i \in \{1, \ldots, n\}$ is not defined or f^A is not defined at $\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n)$. In the first case we conclude that $\mathcal{V}_A(t'_i)$ is also not defined, and in the second case that f^A is not defined at $\mathcal{V}_A(t'_1), \ldots, \mathcal{V}_A(t'_n)$ because $\mathcal{V}_A(t_1) = \mathcal{V}_A(t'_1), \ldots, \mathcal{V}_A(t_n) = \mathcal{V}_A(t'_n)$. This implies that $\mathcal{V}_A(f(t'_1, \ldots, t'_n))$ is not defined. The converse implication follows analogously. Now assume both values are defined and compute

$$\begin{aligned} \mathcal{V}_A(f(t_1, \ldots, t_n)) &= f^A(\mathcal{V}_A(t_1), \ldots, \mathcal{V}_A(t_n)) \\ &= f^A(\mathcal{V}_A(t'_1), \ldots, \mathcal{V}_A(t'_n)) \\ &= \mathcal{V}_A(f(t'_1, \ldots, t'_n)). \end{aligned}$$

This completes the proof. □

If the quotient algebra of Term versus the SIG-congruence defined above is again a model, then it is an initial model.

Theorem 4.3.12 *Let* SPEC $=$ (SIG, E) *be an algebraic specification, and* $\approx$ *the* SIG*-congruence from Theorem 4.3.11. If* Term$/\approx \in$ Gen(SPEC)*, then* Term$/\approx$ *is an initial model.*

Proof. Suppose $A \in$ Gen(SPEC). We have to show that there is a SIG-homomorphism from Term$/\approx$ to A. As in the case of $E = \emptyset$ we use the value function in A and define $h_s([t]) = \mathcal{V}_A(t)$. Notice that h_s is well-defined since $t \sim_s t'$ is equivalent to $\mathcal{V}_A(t) = \mathcal{V}_A(t')$ by the definition of $\sim_s$.

Let $c \in C_s$ be a constant symbol. Then we have $h_s(c^{\text{Term}/\approx}) = h_s([c]) = \mathcal{V}_A(c) = c^A$. Now assume $f \in F_{s_1,\ldots,s_n;s}$ and $[t_1] \in s_1^{\text{Term}/\approx}, \ldots, [t_n] \in s_n^{\text{Term}/\approx}$. Then

$$\begin{aligned} h_s(f^{\text{Term}/\approx}([t_1],\ldots,[t_n])) &= h_s([f^{\text{Term}}(t_1,\ldots,t_n)]) \\ &= h_s([f(t_1,\ldots,t_n)]) \\ &= \mathcal{V}_A(f(t_1,\ldots,t_n)) \\ &= f^A(\mathcal{V}_A(t_1),\ldots,\mathcal{V}_A(t_n)) \\ &= f^A(h_{s_1}([t_1]),\ldots,h_{s_n}([t_n])). \end{aligned}$$

This completes the proof. □

If E is a set of equations, i.e. every formula φ in E is of the form

$$\forall x_1 \in s_1 : \ldots \forall x_n \in s_n : t_{=} t',$$

then we are able to show Term$/\approx \in$ Gen(SPEC). A formal proof of this fact is left as an exercise.

The previous statement can also be formulated as follows: If the set of axioms consists just of equations, then an initial model of the specification exists. It can be computed as quotient algebra of the term algebra.