

Chapter 2

Basic Notions

This chapter provides a brief introduction to basic notions and definitions from algebra and predicate logic. For further discussion, examples, and presentation of concepts see e.g. [55, 22, 169, 106].

2.1 Signatures and Σ -structures

A *signature* constitutes the syntax of a language, i.e. the symbols used to compose language expressions like terms and constraints. Their interpretation or semantics is defined by means of an appropriate *structure*.

Definition 1 (signature) A (many-sorted) **signature** $\Sigma = (S, F, R)$ is defined by a set S of *sorts*, a set F of *function symbols*, and a set R of *predicate symbols*. The sets S , F , and R are mutually disjoint.

Every function symbol $f \in F$ and every predicate symbol $r \in R$ is associated with a *declaration* $f : s_1 \dots s_n \rightarrow s$ and $r : s_1 \dots s_m$, $s, s_i \in S$, $n, m \geq 0$, and thus with an arity n or m , resp. A symbol f with $n = 0$ is a *constant symbol*.

Let X^s be a set of variables of sort $s \in S$. A set of Σ -variables is a set $X = \bigcup_{s \in S} X^s$, where the sets X^s are non-empty and mutually disjoint. \triangleleft

A Σ -*structure* builds on a signature Σ and defines the semantics of the symbols of Σ .

Definition 2 (Σ -structure) Let $\Sigma = (S, F, R)$ be a signature. A Σ -**structure** $\mathcal{D} = (\{\mathcal{D}^s \mid s \in S\}, \{f^{\mathcal{D}} \mid f \in F\}, \{r^{\mathcal{D}} \mid r \in R\})$ consists of an S -sorted set of non-empty *carrier sets* \mathcal{D}^s with $s \in S$, a set of *functions* $f^{\mathcal{D}}$ with $f \in F$, and a set of *predicates* $r^{\mathcal{D}}$ with $r \in R$.

For a function symbol $f \in F$ with $f : s_1 \dots s_n \rightarrow s$ let $f^{\mathcal{D}}$ be an n -ary function, such that $f^{\mathcal{D}} : \mathcal{D}^{s_1} \times \dots \times \mathcal{D}^{s_n} \rightarrow \mathcal{D}^s$ holds.

For a predicate symbol $r \in R$ with $r : s_1 \dots s_m$ let $r^{\mathcal{D}}$ be a m -ary predicate, such that $r^{\mathcal{D}} \subseteq \mathcal{D}^{s_1} \times \dots \times \mathcal{D}^{s_m}$ holds. \triangleleft

The following example illustrates these definitions and will be used in the subsequent sections.

Example 1 Let $\Sigma_{\mathbb{N}} = (S, F, R)$ be a signature consisting of the set of sorts $S = \{\text{nat}\}$, the set $F = \{\text{succ}, \text{plus}, \text{mul}, 0, 1, 2, \dots\}$ of function and constant symbols and the predicate symbols $R = \{\text{eq}, \text{geq}\}$ with the following declarations:

$\text{succ} : \text{nat} \rightarrow \text{nat}$
 $\text{plus}, \text{mul} : \text{nat} \text{ nat} \rightarrow \text{nat}$
 $0, 1, 2, \dots : \text{nat}$
 $\text{eq}, \text{geq} : \text{nat} \text{ nat}$

We define a $\Sigma_{\mathbb{N}}$ -structure $\mathcal{D}_{\mathbb{N}}$ by $\mathcal{D}_{\mathbb{N}} = (\{\mathbb{N}\}, \{f^{\mathbb{N}} \mid f \in F\}, \{r^{\mathbb{N}} \mid r \in R\})$, where the carrier-set \mathbb{N} is the set of natural numbers on which the functions $f^{\mathbb{N}}$ and predicates $r^{\mathbb{N}}$ apply, e.g.

$\text{succ}^{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}$ and for every $x \in \mathbb{N}$ holds: $\text{succ}^{\mathbb{N}}(x) = (x + 1)$,
 $\text{plus}^{\mathbb{N}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, and for every $x, y \in \mathbb{N}$ holds: $\text{plus}^{\mathbb{N}}(x, y) = (x + y)$,

\dots
 $0^{\mathbb{N}} : \mathbb{N}$ with $0^{\mathbb{N}} = 0$,
 $1^{\mathbb{N}} : \mathbb{N}$ with $1^{\mathbb{N}} = 1$,

\dots
 $\text{eq}^{\mathbb{N}} \subseteq \mathbb{N} \times \mathbb{N}$, where for all $x, y \in \mathbb{N}$ holds: $\text{eq}^{\mathbb{N}}(x, y)$ iff $x = y$,
 $\text{geq}^{\mathbb{N}} \subseteq \mathbb{N} \times \mathbb{N}$, where for all $x, y \in \mathbb{N}$ holds: $\text{geq}^{\mathbb{N}}(x, y)$ iff $x \geq y$. \diamond

2.2 Terms, formulae, and validity

Based on the notion of a signature, we define terms and formulae. We provide these syntactic elements with a meaning, i.e. a semantics, and determine the validity of formulae with the help of the corresponding Σ -structure.

In the following, let $\Sigma = (S, F, R)$ be a signature, let X be a set of Σ -variables, and let \mathcal{D} be a Σ -structure.

Terms are built from the symbols of Σ . They are defined inductively. Besides variables and constant symbols, there are terms composed of subterms based on the declarations of the involved function symbols.

Definition 3 (term, ground term) The set $\mathcal{T}(F, X)$ of **terms** over Σ and X is defined as follows: $\mathcal{T}(F, X) = \bigcup_{s \in S} \mathcal{T}(F, X)^s$, where for every sort $s \in S$ the set $\mathcal{T}(F, X)^s$ of terms of sort s is the smallest set containing

1. every variable $x \in X^s$ (of sort s),
2. every 0-ary function symbol $f \in F$ with $f : s$, i.e. every constant symbol, and

3. every expression $f(t_1, \dots, t_n)$, $n \geq 1$, where $f \in F$ is a function symbol with declaration $f : s_1 \dots s_n \rightarrow s$ and every $t_i, i \in \{1, \dots, n\}$, is a (composite) term of $\mathcal{T}(F, X)^{s_i}$.

Terms without variables are **ground terms**. \triangleleft

A *position* p in a term t is represented by a sequence of natural numbers. The empty sequence is denoted by ϵ . We recursively define $t|_p$ to denote the *subterm* of t at position p as $t|_\epsilon = t$ and $f(t_1, \dots, t_n)|_{i.p} = t_i|_p$. By $t[r]_p$ we denote the term which is obtained from t as the result of the *replacement* of the subterm $t|_p$ with the term r .

Example 2 Let $x, y, z \in X$. For our signature $\Sigma_{\mathbb{N}}$ from above, x , 2 , $\text{succ}(x)$, $\text{plus}(2, \text{succ}(3))$, and $\text{plus}(\text{succ}(x), \text{mul}(2, \text{succ}(y)))$ are terms.

For a term $t = \text{plus}(x, \text{mul}(2, \text{succ}(y)))$ examples of subterms are $t|_\epsilon = t$, $t|_1 = x$, $t|_2 = \text{mul}(2, \text{succ}(y))$, $t|_{221} = y$, and replacements are given by e.g. $t[\text{mul}(2, z)]_2 = \text{plus}(x, \text{mul}(2, z))$ and $t[1]_{221} = \text{plus}(x, \text{mul}(2, \text{succ}(1)))$. \diamond

Terms represent elements or objects of the corresponding domain. For example, terms over $\Sigma_{\mathbb{N}}$ are arithmetic expressions. Similarly, boolean expressions can be built over an appropriate signature $\Sigma_{\mathbb{B}}$.

To determine the semantics of terms w. r. t. a Σ -structure \mathcal{D} we must assign values to the variables of the terms. This is done by means of a valuation.

Definition 4 (valuation) An S -sorted family of mappings $\varsigma: X \rightarrow \mathcal{D} = (\varsigma^s : X^s \rightarrow \mathcal{D}^s)_{s \in S}$ which assigns each variable $x \in X^s$ an element of the carrier set \mathcal{D}^s , $s \in S$, is a **valuation**. \triangleleft

Now, we can evaluate terms w. r. t. a structure \mathcal{D} and a valuation ς .

Definition 5 (evaluation of terms) Let $\varsigma: X \rightarrow \mathcal{D}$ be a valuation. The **evaluation** $\tilde{\varsigma}: \mathcal{T}(F, X) \rightarrow \mathcal{D}$ of a term w. r. t. the structure \mathcal{D} and the valuation ς is a family of mappings $(\tilde{\varsigma}^s: \mathcal{T}(F, X)^s \rightarrow \mathcal{D}^s)_{s \in S}$ with:

- $\tilde{\varsigma}^s(x) = \varsigma^s(x)$ for every variable x of sort s ,
- $\tilde{\varsigma}^s(f) = f^{\mathcal{D}}$ for every constant symbol $f \in F$ with $f : s$, and
- $\tilde{\varsigma}^s(f(t_1, \dots, t_n)) = f^{\mathcal{D}}(\tilde{\varsigma}^{s_1}(t_1), \dots, \tilde{\varsigma}^{s_n}(t_n))$ for every function symbol $f \in F$ with $f : s_1 \dots s_n \rightarrow s$ and every sort $s_1, \dots, s_n, s \in S$ and all terms $t_i \in \mathcal{T}(F, X)^{s_i}$, $i \in \{1, \dots, n\}$. \triangleleft

The evaluation of a variable is just its valuation, the evaluation of a constant symbol is the corresponding constant from the structure. For a composite term we evaluate its subterms and apply the corresponding function from the structure.

The set $\text{Formulae}(\Sigma, X)$ of formulae of (first-order) predicate logic determines the *syntax of predicate logic*.

Definition 6 (formulae of predicate logic) The set of **formulae of predicate logic** over a signature Σ and a set of variables X , denoted by $\text{Formulae}(\Sigma, X)$, is inductively defined as follows:

1. For all predicate symbols $r : s_1 \dots s_m$ and all terms $t_i \in \mathcal{T}(F, X)^{s_i}$, $i \in \{1, \dots, m\}$, the expression $r(t_1, \dots, t_m)$ is a (atomic) formula.
2. *true* and *false* are (atomic) formulae.
3. For every formula ϕ the expression $\neg\phi$ is a formula.
4. For all formulae ϕ and ψ the following expressions are formulae too:
 $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \longrightarrow \psi)$, and $(\phi \longleftrightarrow \psi)$.
5. If ϕ is a formula and $x \in X$ is a variable, then $(\forall x.\phi)$ and $(\exists x.\phi)$ are formulae. \triangleleft

We denote the set of variables occurring in a term or formula F , resp., by $\text{var}(F)$. The quantifiers \forall and \exists *bind* variables in formulae. We introduce certain notions concerning quantifiers.

Definition 7 (bound and free variable, open and closed formula) An occurrence of a variable x in a formula $\phi \in \text{Formulae}(\Sigma, X)$ is called **bound**, if x appears in a subformula of ϕ in the form $\exists x.\psi$ or $\forall x.\psi$. Otherwise x is a **free** variable. A formula ϕ without occurrences of free variables is called a **closed** formula, otherwise ϕ is **open**. \triangleleft

Definition 8 (universal and existential closure) Let $\{x_1, \dots, x_n\} \subseteq X$ be the set of free variables of a predicate logic formula $\phi \in \text{Formulae}(\Sigma, X)$. The **universal closure** $\forall\phi$ and the **existential closure** $\exists\phi$ of ϕ are defined by

$$\forall\phi = \forall x_1 \dots \forall x_n. \phi \quad \text{and} \quad \exists\phi = \exists x_1 \dots \exists x_n. \phi, \text{ resp.}$$

The expression \tilde{Y} with $Y \subseteq X$ denotes a (arbitrary) sequence of the variables of the set Y . By $\exists_{\tilde{Y}}\psi$ we denote the existential closure of the formula ψ except for the variables of Y . \triangleleft

In the following, we write $\forall x, y.\phi$ instead of $\forall x.\forall y.\phi$ and $\exists x, y.\phi$ instead of $\exists x.\exists y.\phi$ as is usually done.

Example 3 Consider the signature $\Sigma_{\mathbb{N}}$ and the structure $\mathcal{D}_{\mathbb{N}}$ from Example 1 and the variables $\{x, y, z\} \subset X$.

The following formulae are elements of $\text{Formulae}(\Sigma_{\mathbb{N}}, X)$:
true, *false*, $\text{geq}(2, x)$, $\text{eq}(\text{mul}(2, 2), 4)$, $\neg\text{geq}(2, x) \vee \neg\text{geq}(x, 2)$, $\text{true} \longrightarrow \text{false}$,
 $\text{eq}(2, x) \longleftrightarrow \text{eq}(\text{succ}(2), \text{succ}(x))$, $\forall x, y. \text{eq}(z, \text{plus}(x, y))$, $\forall x. \exists y. \text{eq}(x, \text{succ}(y))$,
 $\text{geq}(x, 2) \longrightarrow \exists x. \text{eq}(x, 2)$.

Consider $p = \forall x, y. \text{eq}(z, \text{plus}(x, y))$ and $q = \text{geq}(x, 2) \longrightarrow \exists x. \text{eq}(x, 2)$. The variables x and y are bound in formula p , while the variable z is free. In the formula q the first occurrence of variable x is free, while its second occurrence is bound by the existential quantifier \exists .

The formulae *true*, *false*, $\text{eq}(\text{mul}(2, 2), 4)$, $\forall x. \exists y. \text{eq}(x, \text{succ}(y))$, and $\text{true} \longrightarrow \text{false}$ are closed, all other formulae given above are open.

Let $Y = \{x, y\} \subseteq \{x, y, z\} \subset X$. The following holds:
 $\exists_{\tilde{Y}} \text{eq}(z, \text{plus}(x, y)) = \exists_{x, y} \text{eq}(z, \text{plus}(x, y)) = \exists z. \text{eq}(z, \text{plus}(x, y)). \quad \diamond$

The *semantics of predicate logic* is determined by the assignment of a meaning to every formula w.r.t. the associated structure. We define the validity relation between structures and formulae (see e.g. [55]).

Definition 9 (validity, \models , model) Let $\phi, \psi \in \text{Formulae}(\Sigma, X)$ be formulae of predicate logic. Let $\varsigma : X \rightarrow \mathcal{D}$ be a valuation. A valuation which maps the variable $x \in X$ to $a \in \mathcal{D}$ and all other variables y to $\varsigma(y)$ is denoted by $\varsigma[x/a] : X \rightarrow \mathcal{D}$, i.e.

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

The relation \models is defined as follows:

$$\begin{aligned} (\mathcal{D}, \varsigma) \models r(t_1, \dots, t_m) & \text{ iff } (\tilde{\varsigma}(t_1), \dots, \tilde{\varsigma}(t_m)) \in r^{\mathcal{D}}, \\ (\mathcal{D}, \varsigma) \models \text{true}, \\ (\mathcal{D}, \varsigma) \not\models \text{false}, \\ (\mathcal{D}, \varsigma) \models \neg\phi & \text{ iff } (\mathcal{D}, \varsigma) \not\models \phi, \\ (\mathcal{D}, \varsigma) \models \phi \wedge \psi & \text{ iff } (\mathcal{D}, \varsigma) \models \phi \text{ and } (\mathcal{D}, \varsigma) \models \psi, \\ (\mathcal{D}, \varsigma) \models \phi \vee \psi & \text{ iff } (\mathcal{D}, \varsigma) \models \phi \text{ or } (\mathcal{D}, \varsigma) \models \psi, \\ (\mathcal{D}, \varsigma) \models \phi \longrightarrow \psi & \text{ iff } (\mathcal{D}, \varsigma) \not\models \phi \text{ or } (\mathcal{D}, \varsigma) \models \psi, \\ (\mathcal{D}, \varsigma) \models \phi \longleftrightarrow \psi & \text{ iff } (\mathcal{D}, \varsigma) \models \phi \longrightarrow \psi \text{ and } (\mathcal{D}, \varsigma) \models \psi \longrightarrow \phi, \\ (\mathcal{D}, \varsigma) \models \forall x. \phi & \text{ iff } (\mathcal{D}, \varsigma[x/a]) \models \phi \text{ for every } a \in \mathcal{D}^s, s \in S, x \in X^s, \\ (\mathcal{D}, \varsigma) \models \exists x. \phi & \text{ iff } (\mathcal{D}, \varsigma[x/a]) \models \phi \text{ for at least one } a \in \mathcal{D}^s, s \in S, x \in X^s. \end{aligned}$$

A formula ϕ is **valid** in \mathcal{D} , i.e. it holds $\mathcal{D} \models \phi$, if for every valuation $\varsigma : X \rightarrow \mathcal{D}$ holds: $(\mathcal{D}, \varsigma) \models \phi$. In this case, we call \mathcal{D} a **model** of ϕ . \triangleleft

Example 4 Consider the signature $\Sigma_{\mathbb{N}}$ and the structure $\mathcal{D}_{\mathbb{N}}$ of Example 1. Let ς be a valuation with $\varsigma(x) = 1$, $\varsigma(y) = 2$, and $\varsigma(z) = 3$. We study the validity of various formulae:

$$\begin{aligned} (\mathcal{D}_{\mathbb{N}}, \varsigma) \models \text{true} \text{ and } (\mathcal{D}_{\mathbb{N}}, \varsigma) \not\models \text{false}. \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \models \text{geq}(2, x), \text{ because } 2^{\mathbb{N}} = 2 \geq 1 = 1^{\mathbb{N}}. \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \models \text{eq}(\text{plus}(2, 2), 4), \text{ because } \text{plus}^{\mathbb{N}}(2, 2) = 4. \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \models \neg \text{geq}(2, x) \vee \neg \text{geq}(x, 2), \\ \quad \text{because the validity of one subformula } \phi \text{ or } \psi \text{ of a formula } \phi \vee \psi \text{ is} \\ \quad \text{sufficient and } (1, 2) \notin \text{geq}^{\mathbb{N}} \text{ resp. } 1 \not\geq 2. \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \not\models \text{true} \longrightarrow \text{false} \\ \quad \text{according to the definition of validity of formulae of the form } \phi \longrightarrow \psi \\ \quad \text{(see above).} \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \models (\text{eq}(2, x) \longleftrightarrow \text{eq}(\text{succ}(2), \text{succ}(x))), \\ \quad \text{because } (2, 1) \notin \text{eq}^{\mathbb{N}} \text{ resp. } 2 \neq 1 \text{ and } (3, 2) \notin \text{eq}^{\mathbb{N}} \text{ resp. } 3 \neq 2. \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \not\models \forall x, y. \text{eq}(z, \text{plus}(x, y)), \\ \quad \text{because there are valuations } \varsigma' \text{ of } x \text{ and } y \text{ such that } 3 \neq \varsigma'(x) + \varsigma'(y). \\ (\mathcal{D}_{\mathbb{N}}, \varsigma) \not\models \forall x. \exists y. \text{eq}(x, \text{succ}(y)), \end{aligned}$$

because when x has the value 0 there is no value for y such that $x = y + 1$.

$(\mathcal{D}_{\mathbb{N}}, \varsigma) \models \text{geq}(x, 2) \longrightarrow \exists x. \text{eq}(x, 2)$, because $(1, 2) \notin \text{geq}^{\mathbb{N}}$.

Of the above formulae the following are valid in $\mathcal{D}_{\mathbb{N}}$, i. e. they hold in $\mathcal{D}_{\mathbb{N}}$ for every valuation: true , $\text{eq}(\text{plus}(2, 2), 4)$, $\text{eq}(2, x) \longleftrightarrow \text{eq}(\text{succ}(2), \text{succ}(x))$, and $\text{geq}(x, 2) \longrightarrow \exists x. \text{eq}(x, 2)$. \diamond

2.3 Substitutions and unifiers

When defining operational principles of programming languages later on in this book, we will need certain notions concerning substitutions.

A *substitution* applied to a term or atomic formula replaces variables by terms.

Definition 10 (substitution) A **substitution** σ is a function $\sigma : X \rightarrow \mathcal{T}(F, X)$ with $\sigma(x) \in \mathcal{T}(F, X)^s$ for all $x \in X^s$.

We extend the function σ to $\tilde{\sigma} : \mathcal{T}(F, X) \rightarrow \mathcal{T}(F, X)$, i. e. for application on terms by

- $\tilde{\sigma}(x) = \sigma(x)$ for all variables $x \in X$,
- $\tilde{\sigma}(f(t_1, \dots, t_n)) = f(\tilde{\sigma}(t_1), \dots, \tilde{\sigma}(t_n))$ for all terms $f(t_1, \dots, t_n)$.

Analogously, σ is extended for application on atomic formulae. In the following, we identify a substitution σ with its extension $\tilde{\sigma}$ and write σ instead of $\tilde{\sigma}$. \triangleleft

In this book, we deal with *finite* substitutions σ in the sense that for only finitely many variables x holds: $\sigma(x) \neq x$. A substitution σ can, thus, be represented in the form $\sigma = \{x/\sigma(x) \mid \sigma(x) \neq x\}$, where we explicitly enumerate all its elements. We denote the identity substitution by *id*.

The *composition* of substitutions describes the sequential application of substitutions on a term or formulae.

Definition 11 (composition of substitutions) The **composition** of substitutions σ and ϕ is defined by $(\phi \circ \sigma)(e) = \phi(\sigma(e))$ for all terms and atomic formulae e . \triangleleft

Example 5 Consider a set X of variables with $\{x, y, z\} \subseteq X$ and the signature $\Sigma_{\mathbb{N}}$ from Example 1.

Let σ and ϕ be substitutions with $\sigma = \{x/4, y/\text{plus}(3, z)\}$ and $\phi = \{z/1\}$. The following holds:

$$\begin{aligned} \sigma(\text{succ}(2)) &= \text{succ}(\sigma(2)) = \text{succ}(2) \\ \sigma(\text{succ}(x)) &= \text{succ}(\sigma(x)) = \text{succ}(4) \\ \sigma(\text{mul}(3, \text{plus}(x, \text{succ}(y)))) &= \text{mul}(3, \text{plus}(4, \text{succ}(\text{plus}(3, z)))) \end{aligned}$$

$$\begin{aligned}
(\phi \circ \sigma)(mul(3, plus(x, succ(y)))) &= \phi(\sigma(mul(3, plus(x, succ(y))))) \\
&= \phi(mul(3, plus(4, succ(plus(3, z))))) \\
&= mul(3, plus(4, succ(plus(3, 1)))) \quad \diamond
\end{aligned}$$

Unifiers are substitutions which allow one to identify certain terms or formulae.

Definition 12 (unifier, most general unifier, *mgu*) Let s and t be terms or atoms. A substitution σ with $\sigma(s) = \sigma(t)$ is a **unifier** of s and t . A unifier σ of s and t is a **most general unifier** (we write $\sigma = mgu(s, t)$) if for every unifier ϕ of s and t there exists a substitution ψ such that $\phi = \psi \circ \sigma$ holds. \triangleleft

For algorithms to compute most general unifiers we refer to [194, 106]. If two terms or atoms s and t are not unifiable, we write $mgu(s, t) = \emptyset$.¹

Example 6 Consider the signature $\Sigma_{\mathbb{N}}$ and the set $\{x, y, z\} \subseteq X$. The terms $s = mul(x, succ(z))$ and $t = mul(2, y)$ are unifiable with substitution $\sigma = \{x/2, y/succ(z)\}$, i.e. σ is a unifier of s and t :

$$\sigma(s) = mul(2, succ(z)) = \sigma(t)$$

The substitution $\phi = \{x/2, y/succ(3), z/3\}$ is a unifier of s and t too:

$$\phi(s) = mul(2, succ(3)) = \phi(t)$$

The substitution σ is a most general unifier of s and t . For σ and ϕ there is a substitution $\psi = \{z/3\}$ such that $\phi = \psi \circ \sigma$ holds. \diamond

Let the *parallel composition* of idempotent substitutions be defined as in [175]. We compute the parallel composition $(\sigma \uparrow \phi)$ of two idempotent substitutions σ and ϕ as follows:

$(\sigma \uparrow \phi) = mgu(f(x_1, \dots, x_n, y_1, \dots, y_m), f(\sigma(x_1), \dots, \sigma(x_n), \phi(y_1), \dots, \phi(y_m)))$, where $x_i, i \in \{1, \dots, n\}$, and $y_j, j \in \{1, \dots, m\}$, are the domain variables of σ and ϕ , resp.

Example 7 Given $\Sigma_{\mathbb{N}}$, a set X of variables with $\{w, x, y, z\} \subseteq X$, and the substitutions $\sigma = \{x/0, y/z\}$, $\phi = \{w/succ(x), y/0\}$, and $\psi = \{y/0, z/succ(0)\}$, we build their parallel compositions:

$$\begin{aligned}
(\sigma \uparrow \phi) &= mgu(f(x, y, w, y), f(0, z, succ(x), 0)) \\
&= \{x/0, y/0, z/0, w/succ(0)\} \\
(\sigma \uparrow \psi) &= mgu(f(x, y, y, z), f(0, z, 0, succ(0))) = \emptyset \\
(\phi \uparrow \psi) &= mgu(f(w, y, y, z), f(succ(x), 0, 0, succ(0))) \\
&= \{w/succ(x), y/0, z/succ(0)\} \quad \diamond
\end{aligned}$$

¹ Note that \emptyset has a completely different meaning than the identity substitution *id*.



<http://www.springer.com/978-3-642-17329-5>

Multiparadigm Constraint Programming Languages

Hofstedt, P.

2011, XII, 180 p., Hardcover

ISBN: 978-3-642-17329-5