On the algebra of structured specifications

Răzvan Diaconescu

Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Romania

Ionuţ Ţuţu

Şcoala Normală Superioară Bucureşti, Romania

Abstract

We develop module algebra for structured specifications with model oriented denotations. Our work extends the existing theory with specification building operators for non-protecting importation modes and with new algebraic rules (most notably for initial semantics) and upgrades the pushout-style semantics of parameterized modules to capture the (possible) sharing between the body of the parameterized modules and the instances of the parameters. We specify a set of sufficient abstract conditions, smoothly satisfied in the actual situations, and prove the isomorphism between the parallel and the serial instantiation of multiple parameters. Our module algebra development is done at the level of abstract institutions, which means that our results are very general and directly applicable to a wide variety of specification and programming formalisms that are rigorously based upon some logical system.

1. Introduction

It is a great honour for us to dedicate this work to Professor Jan Bergstra on the occasion of his 60th birthday. The significance of his many seminal contributions to theoretical computing science has not only influenced the education and the way of thinking of several generations of researchers in the area, but also goes beyond computing science, some of them touching fundamental aspects of our basic mathematical education and preconceptions [4]. The style of his scientific contributions is based upon the use of (general) algebra in a rather clear and elegant way, an aspect that unfortunately is the exception rather than the rule within the current research activities in computing science.

Our paper is related to the seminal work of Professor Jan Bergstra and his collaborators on module algebra [3]. To our understanding, [3] was the true start of the concept of module algebra as the study of the algebraic rules satisfied by the module expressions of a software system (especially specification and programming) that employs a well-developed structuring mechanism. In module algebra it is also important that the software system is rather rigorously based upon a logical system, for example, many sorted (classical) first order logic in the case of [3]. Let us refrain from repeating here the well-known arguments showing the crucial role played by modularization or structuring of system specifications or of programs, a paradigm sometimes referred to as specification or programming 'in-the-large'. Instead, let us mention that module algebra has several important consequences including crucial support for evaluation of module expressions and for specification and programming in-the-large methodologies.

Email addresses: Razvan.Diaconescu@imar.ro (Răzvan Diaconescu), ittutu@gmail.com (Ionuţ Ţuţu)

The first author of this paper (abbreviated RD) got involved with module algebra in 1991 when he was a DPhil student at Oxford. His Professor, the late Joseph Goguen, an emblematic figure in many areas of science and one of the most elegant promoters of the algebraic methods in computing science, had at that moment invited RD to join his project aimed to give a replica of [3] in line with the formal specification trend of developing things independently of any concrete underlying logic, that is of doing module algebra within the abstract institution theory of Goguen and Burstall [16]. The result of this research project was reported in the paper [14], which may be still one of the most cited scientific publications of RD, and which introduced or shed a new light on several theoretical concepts that have influenced much of the work in the area. Three of the main achievements of [14] that can be noticed now after 20 years are

1. the category theoretic concept of 'inclusion system',
2. the light shed on the importance of a model amalgamation property, called '(semi-)exactness', in structuring of specifications, and
3. the beginning of a long process of understanding at a general abstract level of what form of interpolation property is needed for the underlying logic to support a well-behaved module system.

All above mentioned achievements which are reflected in a great deal of work developed by many researchers worldwide, and that would take too much space to cite here, have [3] as one of their main causes.

After [14] had been completed, before it was published, we (Professor Joseph Goguen and RD) paid a visit to Professor Bergstra's group in Amsterdam to discuss and compare the two perspectives on module algebra. As a DPhil student I (RD) had very little understanding of real computing science issues, and now I recall quite vividly how interesting was the dialogue between Professor Bergstra and Professor Goguen. There were very few moments in my development as a scientist of such learning intensity. After that I and Professor Bergstra met in person only a couple of times, and in all those occasions I felt much friendship and encouragement from Professor Bergstra.

1.1. The structure and contributions of this paper

There are at least three levels of giving semantics or denotations to specification modules:

1. The set of the (accumulated) axioms of the specification; this underlies [3] and is the most syntactic level. The paper [14] argues about several shortcomings of this viewpoint.
2. The theory of the specification, in the sense of the closure of its set of axioms under (semantic) deduction. This point of view is taken by [14] and can be considered as a middle grained approach.
3. The class of models of the specification in the style of [28, 6]. This is the most semantic approach.

The specification theory literature contains many arguments in favour of each of these viewpoints on denotations for specification modules, and even many more arguments against each of them.

A good way to view these different perspectives on the semantics of specification modules is that these refer to slightly different aspects of the same phenomena. However, we can now understand that since the former two approaches are *property oriented*, in the sense that their emphasis is on the properties satisfied by the models of the specifications, they do not consider an important point of structuring which is realized only by the latter *model oriented* approach. Everybody sees specification or programming in-the-large as the only realistic way to build complex specifications or programs; however, many neglect another important motivation, namely that of being able to specify classes of models much beyond the capabilities of specification in-the-small. A good example illustrating this difference in specification power is given by fields.

One can actually prove that the class of fields does not admit a specification in-the-small within any form of Horn clause logic (including even the use of partial functions); however, it admits (see Example 3.1 below) a rather simple specification in-the-large in order sorted (unconditional) equational logic, or equivalently in many sorted conditional equational logic.[1]

While we think that the study of module algebra is important for all of the three above mentioned approaches, here (unlike in [14]) we focus on the development of module algebra for the model oriented one. This is done at the generic level of abstract institutions independently of any concrete underlying logical system (like in [14], in the style introduced by [16]), our work being thus directly applicable to a multitude of specification formalisms based upon various logics.

Our paper is structured as follows.

1. The first preliminary section introduces the basic category and institution theoretical concepts necessary for our work. This includes a series of new developments (concepts and results) on the theory of inclusion systems that are required especially by the part on parameterization.

2. In the next section in addition to recalling the standard primitive specification building operators from the literature we also introduce new ones and extend some of the established ones in connection to non-protecting importation modes. (By non-protecting importation modes we mean imports that are not required to satisfy what is commonly known as the 'no-junk' and 'no-confusion' conditions.) As far as we are aware this is the first module algebra study that includes the phenomenon of non-protecting importation modes. Here we also recall some established basic concepts and results from the theory of structured specifications at the level of abstract institutions but also develop some new ones.

3. The third technical section is devoted to the study of the algebraic rules for our specification building operators that are satisfied by the model oriented denotations of structured specifications. Here besides recalling important known rules we also study novel rules, some of them related to our new operators (about non-protecting modes). An important class of new rules studied here are those concerning the initial semantics operator, such as its distributivity over module sums.

4. In the last technical section we develop a semantics for the so-called pushout-style parameterization (à la Clear [7]) at the level of abstract institutions that upgrades the existing one to capture the sharing between the body of the parameterized module and the instance of the parameter. This relies crucially upon our use of inclusion systems for structuring specifications, and we think that the resulting theory captures most realistically the actual practice of pushout-style parameterized specifications. The section ends with the proof of a rule of the form

$$SP(P_1 \cup P_2 \Leftarrow v_1 + v_2) \cong SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$$

that expresses the isomorphism between the simultaneous (parallel) and the sequential (serial) instantiation of multiple parameters. This rather mathematically difficult result relies upon the capture of a set of sufficient abstract conditions for the underlying institution that are smoothly satisfied in the actual situations.

[1] The elimination of the partiality of division by 0 has been an important research project of Professor Bergstra in the recent years [4].

2. Preliminaries

The aim of this section is to introduce the category and institution theory concepts and notations necessary for our work. An important part of this section is devoted to the development of a series of new technical concepts and results about inclusion systems.

2.1. Categories

We assume the reader is familiar with basic notions and standard notations from category theory; e.g., see [22] for an introduction to this subject. With respect to notational conventions, $|\mathbb{C}|$ denotes the class of objects of a category $\mathbb{C}$, $\mathbb{C}(A, B)$ the set of arrows (morphisms) with domain A and codomain B, and composition is denoted by ";" and in diagrammatic order. The category of sets (as objects) and functions (as arrows) is denoted by $\mathbb{S}et$, and $\mathbb{CAT}$ is the category of all categories.[2]

Given $\mathcal{H}_1, \mathcal{H}_2 \subseteq \mathbb{C}$ by $\mathcal{H}_1; \mathcal{H}_2$ we denote the class of arrows $\{h_1; h_2 \mid h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2\}$. Given $\mathcal{H} \in \mathbb{C}$ by $\overset{\mathcal{H}}{\rightarrow}$ we denote the binary relation on $|\mathbb{C}|$ given by $A \overset{\mathcal{H}}{\rightarrow} B$ if and only if there exists $(h\colon A \rightarrow B) \in \mathcal{H}$. Also $\overset{\mathcal{H}}{\leftarrow}$ denotes the inverse of $\overset{\mathcal{H}}{\rightarrow}$.

2.2. Institutions

Institutions have been defined by Goguen and Burstall in [8], the seminal paper [16] being printed after a delay of many years. Below we recall the concept of institution which formalizes the intuitive notion of logical system, including syntax, semantics, and the satisfaction between them.

Definition 2.1 (Institutions)**.** An *institution* $\mathcal{I} = (\mathbb{S}ig^{\mathcal{I}}, \mathsf{Sen}^{\mathcal{I}}, \mathsf{Mod}^{\mathcal{I}}, \models^{\mathcal{I}})$ consists of

1. a category $\mathbb{S}ig^{\mathcal{I}}$, whose objects are called *signatures*,
2. a functor $\mathsf{Sen}^{\mathcal{I}}\colon \mathbb{S}ig^{\mathcal{I}} \rightarrow \mathbb{S}et$, giving for each signature a set whose elements are called *sentences* over that signature,
3. a functor $\mathsf{Mod}^{\mathcal{I}}\colon (\mathbb{S}ig^{\mathcal{I}})^{\mathrm{op}} \rightarrow \mathbb{CAT}$ giving for each signature Σ a category whose objects are called Σ-*models*, and whose arrows are called Σ-(*model*) *morphisms*, and
4. a relation $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathsf{Mod}^{\mathcal{I}}(\Sigma)| \times \mathsf{Sen}^{\mathcal{I}}(\Sigma)$ for each $\Sigma \in |\mathbb{S}ig^{\mathcal{I}}|$, called Σ-*satisfaction*,

such that for each morphism $\varphi\colon \Sigma \rightarrow \Sigma'$ in $\mathbb{S}ig^{\mathcal{I}}$, the *satisfaction condition*

$$M' \models^{\mathcal{I}}_{\Sigma'} \mathsf{Sen}^{\mathcal{I}}(\varphi)(\rho) \quad \text{if and only if} \quad \mathsf{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho$$

holds for each $M' \in |\mathsf{Mod}^{\mathcal{I}}(\Sigma')|$ and $\rho \in \mathsf{Sen}^{\mathcal{I}}(\Sigma)$.

We denote the *reduct* functor $\mathsf{Mod}^{\mathcal{I}}(\varphi)$ by $\_{\restriction_\varphi}$ and the sentence translation $\mathsf{Sen}^{\mathcal{I}}(\varphi)$ by $\varphi(\_)$. When $M = M'{\restriction_\varphi}$ we say that M is a φ-*reduct of* M', and that M' is a φ-*expansion of* M. When there is no danger of ambiguity, we may skip the superscripts from the notations of the entities of the institution; for example $\mathbb{S}ig^{\mathcal{I}}$ may be simply denoted $\mathbb{S}ig$.

Notation 2.1. In any institution as above we use the following notations:

– for any $\mathcal{M} \subseteq |\mathsf{Mod}(\Sigma)|$, $\mathcal{M}^*$ denotes $\{\rho \in \mathsf{Sen}(\Sigma) \mid M \models_\Sigma \rho \text{ for each } M \in \mathcal{M}\}$.

– for any $E \subseteq \mathsf{Sen}(\Sigma)$, E^* denotes $\{M \in |\mathsf{Mod}(\Sigma)| \mid M \models_\Sigma \rho \text{ for each } \rho \in E\}$.

[2] Strictly speaking, this is only a quasi-category living in a higher set-theoretic universe.

– for any $E, E' \subseteq \mathsf{Sen}(\Sigma)$, $E \models E'$ denotes $E^* \subseteq E'^*$.

– for any $E \subseteq \mathsf{Sen}(\Sigma)$, $\mathsf{Mod}(\Sigma, E)$ is the full subcategory of $\mathsf{Mod}(\Sigma)$ whose objects are in E^*.

Definition 2.2 (Preservation of Sentences)**.** In an institution, given a class $\mathcal{H}$ of model homomorphisms, we say that $\mathcal{H}$ *preserves the satisfaction of a sentence* ρ when $M \models \rho$ and $M \xrightarrow{\mathcal{H}} N$ implies $N \models \rho$.

General assumption: We assume that model isomorphisms preserve the satisfaction of all sentences of the institutions. It is easy to see that this assumption holds in all the concrete examples of institutions of interest for specification and programming.

There is a myriad of examples of logics captured as institutions, both from logic and computing. A few of them can be found in [12]. In fact the thesis underlying institution theory is that anything that deserves to be called logic can be captured as institution. Let us very briefly present only the following example.

Example 2.1 (Algebra (**MSA**, **OSA**))**.** The *many sorted algebra* (**MSA**) *signatures* are pairs (S, F) consisting of a set of sort symbols S and of a family $F = \{F_{w \to s} \mid w \in S^*, s \in S\}$ of sets of function symbols indexed by arities (for the arguments) and sorts (for the results). *Signature morphisms* $\varphi\colon (S, F) \to (S', F')$ consist of a function $\varphi^{\mathrm{st}}\colon S \to S'$ and a family of functions $\varphi^{\mathrm{op}} = \{\varphi^{\mathrm{op}}_{w \to s}\colon F_{w \to s} \to F'_{\varphi^{\mathrm{st}}(w) \to \varphi^{\mathrm{st}}(s)} \mid w \in S^*, s \in S\}$.

The (S, F)-*models* M, called algebras, interpret each sort symbol s as a set M_s and each function symbol $\sigma \in F_{w \to s}$ as a function M_σ from the product M_w of the interpretations of the argument sorts to the interpretation M_s of the result sort. An (S, F)-*model homomorphism* $h\colon M \to M'$ is an indexed family of functions $\{h_s\colon M_s \to M'_s\}_{s \in S}$ such that $h_s(M_\sigma(m)) = M'_\sigma(h_w(m))$ for each $\sigma \in F_{w \to s}$ and each $m \in M_w$ where $h_w\colon M_w \to M'_w$ is the canonical component-wise extension of h, i.e. $h_w(m_1, \ldots, m_n) = (h_{s_1}(m_1), \ldots, h_{s_n}(m_n))$ for $w = s_1 \ldots s_n$ and $m_i \in M_{s_i}$.

For each signature morphism φ, the *reduct* $M' \restriction_\varphi$ of a model M' is defined by $(M' \restriction_\varphi)_x = M'_{\varphi(x)}$ for each sort or function symbol x from the domain signature of φ.

Sentences are the usual first order sentences built from equational atoms $t = t'$, with t and t' (well-formed) terms of the same sort, by iterative application of Boolean connectives and quantifiers. Sentence translations along signature morphisms just rename the sorts and function symbols according to the respective signature morphisms. They can be formally defined by induction on the structure of the sentences.

The satisfaction of sentences by models is the usual Tarskian satisfaction defined inductively on the structure of the sentences.

OSA [18, 17] refines **MSA** by considering a partial order structure on the sets of sorts of a signature, which at the semantics level is reflected as a set theoretic inclusion between the corresponding carriers. Therefore **OSA** *signatures* are tuples $(S, \leq, F)$ such that (S, F) is a **MSA** signature and $\leq$ is a partial order on S satisfying the following monotonicity condition: for any operation symbol $\sigma \in F_{w_1 \to s_1} \cap F_{w_2 \to s_2}$, if $w_1 \leq w_2$ then $s_1 \leq s_2$. *Signature morphisms* $\varphi\colon (S, \leq, F) \to (S', \leq', F')$ are **MSA** signature morphisms such that the sort component $\varphi^{\mathrm{st}}\colon (S, \leq) \to (S', \leq')$ is an order-preserving function.

The **OSA** *models* M, or order sorted algebras, of a given **OSA** signature $(S, \leq, F)$ are (S, F)-algebras satisfying the following two monotonicity conditions:

1. $M_{s_1} \subseteq M_{s_2}$, whenever $s_1 \leq s_2$, and
2. $M_{\sigma\colon w_1 \to s_1}$ and $M_{\sigma\colon w_2 \to s_2}$ agree on M_{w_1}, whenever $w_1 \leq w_2$ and $\sigma \in F_{w_1 \to s_1} \cap F_{w_2 \to s_2}$.

An $(S, \leq, F)$-*morphism* $h\colon M \to M'$ is an (S, F)-morphism such that for any two sorts s_1 and s_2, if $s_1 \leq s_2$ then h_{s_1} and h_{s_2} agree on M_{s_1}.

Both *sentences* and *satisfaction* are defined as in the case of **MSA** with the observation that for any two sorts s_1 and s_2 such that $s_1 \leq s_2$, the well-formed terms of sort s_1 are also well-formed terms of sort s_2.

2.3. Model amalgamation

The crucial role of model amalgamation for the semantics studies of formal specifications comes up in a lot of works in the area, a few early examples being [28, 29, 23, 14]. The model amalgamation property is a necessary condition in many institution-independent model theoretic results (see [12]), thus being one of the most desirable properties for an institution. It can be considered even as more fundamental than the satisfaction condition since in institutions with quantifications it is used in one of its weak forms in the proof of the satisfaction condition at the induction step corresponding to quantifiers. Its importance within the context of module algebra has been first emphasized in [14]. Model amalgamation properties for institutions formalize the possibility of amalgamating models of different signatures when they are consistent on some kind of generalized 'intersection' of signatures.

Definition 2.3 (Model Amalgamation)**.** A commutative square of signature morphisms

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\theta_1} \\ \Sigma_2 & \xrightarrow[\theta_2]{} & \Sigma' \end{array}$$

is a *weak amalgamation square* if and only if for each Σ_1-model M_1 and Σ_2-model M_2 such that $M_1{\restriction_{\varphi_1}} = M_2{\restriction_{\varphi_2}}$, there exists a Σ'-model M' such that $M'{\restriction_{\theta_1}} = M_1$ and $M'{\restriction_{\theta_2}} = M_2$.

It is a *model amalgamation square* when in addition M' is unique; in such a case M' may be denoted $M_1 \otimes_{\varphi_1,\varphi_2} M_2$, or $M_1 \otimes M_2$ for short when there is no danger of ambiguity.

In most of the institutions formalizing conventional or non-conventional logics, pushout squares of signature morphisms are model amalgamation squares [14, 12]. These of course include our benchmark **MSA** example.

Definition 2.4. An institution *has (weak) model amalgamation* when each pushout square of signatures is a (weak) amalgamation square.

A *semi-exact institution* is an institution with the model amalgamation property extended also to model homomorphisms, or equivalently an institution with a model functor $\mathsf{Mod}^{\mathcal{I}} : (\mathbb{S}ig^{\mathcal{I}})^{\mathrm{op}} \to \mathbb{CAT}$ that preserves pullbacks.

2.4. Inclusion systems

Inclusion systems were introduced in [14] as a categorical device supporting an abstract general study of structuring of specification and programming modules that is independent of any underlying logic. They have been used in a series of general module algebra studies such as [14, 19, 12] but also for developing axiomatizability [26, 10, 12] and definability [2] results within the framework of the so-called 'institution-independent model theory' [12]. Inclusion systems capture categorically the concept of set-theoretic inclusion in a way reminiscent of how the rather notorious concept of factorization system [5] captures categorically the set-theoretic injections; however in many applications the former are more convenient than the latter. Here we first recall from the literature the basics of the theory of inclusion systems and after we develop a series of new concepts and results needed by our work here.

The definition below can be found in the recent literature on inclusion systems (e.g. [12]) and differs slightly from the original one of [14].

Definition 2.5 (Inclusion Systems)**.** $\langle \mathcal{I}, \mathcal{E} \rangle$ is a *inclusion system* for a category $\mathbb{C}$ if $\mathcal{I}$ and $\mathcal{E}$ are two subcategories with $|\mathcal{I}| = |\mathcal{E}| = |\mathbb{C}|$ such that

1. $\mathcal{I}$ is a partial order (with the ordering relation denoted by $\subseteq$), and
2. every arrow f in $\mathbb{C}$ can be factored uniquely as $f = e_f; i_f$ with $e_f \in \mathcal{E}$ and $i_f \in \mathcal{I}$.

The arrows of $\mathcal{I}$ are called *abstract inclusions*, and the arrows of $\mathcal{E}$ are called *abstract surjections*. The domain of the inclusion i_f in the factorization of f is called the *image of* f and is denoted as $\mathrm{Im}(f)$ or $f(A)$ when A is a domain of f. An inclusion $i\colon A \to B$ may be also denoted simply by $A \subseteq B$.

The inclusion system

– is *epic* when all abstract surjections are epis,

– has *unions* when $\mathcal{I}$ has finite least upper bounds (denoted $\cup$),

– has *intersections* when $\mathcal{I}$ has greatest lower bounds (denoted $\cap$), and

– is *distributive* when it has unions and intersections that satisfy the usual distributivity rules.

In [9] it is shown that the class $\mathcal{I}$ of the abstract inclusions determines the class $\mathcal{E}$ of the abstract surjections. In this sense, [9] gives an explicit equivalent definition of inclusion systems which uses only the class $\mathcal{I}$ of the abstract inclusions. In [14] it has been shown that whenever the category $\mathbb{C}$ has pullbacks the existence of unions implies the existence of the intersections that are obtained as pullbacks of unions.

$$\begin{array}{ccc} A \cap B & \xrightarrow{\subseteq} & A \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} \\ B & \xrightarrow[\subseteq]{} & A \cup B \end{array}$$

Whenever we use unions and intersections we implicitly assume that the considered inclusion system has them. It is often useful that the intersection-union squares are not only pullbacks, but they are also pushouts. Although this property is widely spread among inclusion systems of interest, it does not hold in general and therefore at the level of abstract inclusion systems it has to be assumed when necessary.

The standard example of inclusion system is that from $\mathbb{S}et$, with set theoretic inclusions in the role of the abstract inclusions and the surjective functions in the role of the abstract surjections. It is easy to note that this has all properties introduced by Definition 2.5 above. The literature contains myriads of examples of inclusion systems for categories of signatures and for categories of models of various institutions from logic or from specification theory. Due to lack of space let us here recall only a couple of the most representative ones, examples of great significance for our work here.

Example 2.2 (Inclusion Systems for **MSA** Signatures)**.** Besides the trivial inclusion system that can be defined in any category (i.e. identities as abstract inclusions and all arrows as abstract surjections) the category of the **MSA** signatures admits also the following non-trivial inclusion systems:

Inclusion system	Abstract surjections $\varphi\colon (S,F) \to (S',F')$	Abstract inclusions $(S,F) \subseteq (S',F')$
Closed	$\varphi^{\mathrm{st}}\colon S \to S'$ surjective	$S \subseteq S'$ $F_{w\to s} = F'_{w\to s}$ for $w \in S^*$, $s \in S$
Strong	$\varphi^{\mathrm{st}}\colon S \to S'$ surjective $F'_{w'\to s'} = \bigcup_{\varphi^{\mathrm{st}}(ws)=w's'} \varphi^{\mathrm{op}}(F_{w\to s})$	$S \subseteq S'$ $F_{w\to s} \subseteq F'_{w\to s}$ for $w \in S^*$, $s \in S$

Note that the strong inclusion systems for the **MSA** signatures is epic and is distributive (which implies it has unions and intersections) while the closed one has none of these properties.

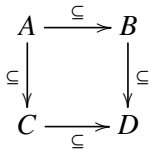
The following abstract concept that captures a rather common situation in practice, including of course **MSA**, has been introduced in [14].

Definition 2.6 (Inclusive Institutions)**.** An institution is inclusive when its category of signatures is endowed with an inclusion system such that whenever $\Sigma \subseteq \Sigma'$ we have $\mathsf{Sen}(\Sigma) \subseteq \mathsf{Sen}(\Sigma')$.

For this work we assume the institutions to be inclusive.

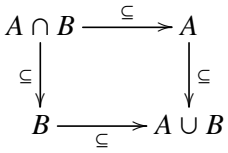
In the following we introduce some new concepts and develop new results about inclusion systems that are necessary for our work here.

Fact 2.1. *If a square of inclusions like below is a pushout then $D = B \cup C$.*



The following abstracts the concept of disjointness from sets and **MSA** signatures to abstract inclusion systems. Then it can be instantiated to many concrete frameworks.

Definition 2.7 (Disjoint Objects)**.** In a category with pullbacks and a designated inclusion system we say that two objects A and B are *disjoint* if and only if the intersection-union square



describes a pushout and $A \cap B$ is an initial object in the category.

Example 2.3. Note that the concept of disjoint objects in $\mathbb{S}et$ just means ordinary disjoint sets, while two signatures (S_1, F_1) and (S_2, F_2) are disjoint (with respect to the strong inclusion system for the **MSA** signatures) if and only if $S_1 \cap S_2 = \emptyset$. If we considered *single sorted* signatures then disjointness of signatures F_1 and F_2 means $(F_1)_n \cap (F_2)_n = \emptyset$ for each arity $n \in \omega$.

Directly from Definition 2.7 by the well-known expression of coproducts as pushouts we obtain the following.

Corollary 2.1. *If A and B are disjoint then $A \cup B$ is the coproduct of A and B.*

Proposition 2.1. *If $B' \subseteq B$ and A and B are disjoint, then A and B' are disjoint too.*

Proof. Since the intersection is the infimum with respect to the partial order given by the inclusions, we have that $A \cap B' \subseteq A \cap B$. By hypothesis $A \cap B$ is initial, thus there exists an unique arrow $f : A \cap B \to A \cap B'$. By the uniqueness feature of the factorization of $1_{A \cap B}$ it follows that f is inclusion and hence $A \cap B = A \cap B'$ which means $A \cap B'$ is initial. □

Proposition 2.2. *In any distributive inclusion system if A and C are disjoint and B and C are disjoint then $A \cup B$ and C are disjoint.*

Proof. By distributivity we have that $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$. By the disjointness hypotheses we have that both $A \cap C$ and $B \cap C$ are initial. By Proposition 2.1 twice we have that $A \cap C$ and $B \cap C$ are disjoint; therefore, by Corollary 2.1, we obtain that $(A \cap C) \cup (B \cap C)$ is the coproduct of two initial objects and hence it is initial too. We have thus shown that $(A \cup B) \cap C$ is initial. □

The following generalizes the concept of compatible signature morphisms from the language CASL [24] to abstract inclusion systems.

Definition 2.8 (Compatible Arrows)**.** Two arrows $f_1 \colon A_1 \to B$ and $f_2 \colon A_2 \to B$ are *compatible* when $(A_1 \cap A_2 \subseteq A_1); f_1 = (A_1 \cap A_2 \subseteq A_2); f_2$.

Notation 2.2. If the intersection-union square below is a pushout square

$$\begin{array}{ccc} A_1 \cap A_2 & \xrightarrow{\subseteq} & A_1 \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} \\ A_2 & \xrightarrow[\subseteq]{} & A_1 \cup A_2 \end{array}$$

then for any two compatible arrows $f_1 \colon A_1 \to B$ and $f_2 \colon A_2 \to B$ we denote by $f_1 \vee f_2$ the unique arrow $A_1 \cup A_2 \to B$ such that $(A_i \subseteq A_1 \cup A_2); (f_1 \vee f_2) = f_i$ for $i \in \{1, 2\}$.

Proposition 2.3. *In a category endowed with an inclusion system that has unions and intersections we assume the following:*

1. *each intersection-union square is a pushout square, and*
2. *A and A' are two objects such that $A \subseteq A'$.*

In the commutative diagram below, the right-hand square $[A \cup B, A' \cup B, B, B']$ describes a pushout if and only if the outer square $[A \cup (B \cap A'), A', B, B']$ describes a pushout.

$$\begin{array}{ccccc} A \cup (B \cap A') & \xrightarrow{\subseteq} & A \cup B & \xrightarrow{v} & B \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} & & \downarrow{\scriptstyle f} \\ A' & \xrightarrow[\subseteq]{} & A' \cup B & \xrightarrow[v']{} & B' \end{array}$$

Proof. By the general result saying that gluing together pushout squares yields a pushout square it is enough if we showed that the left-hand square $[A \cup (B \cap A'), A', A \cup B, A' \cup B]$ depicts a pushout. In order to show this we glue another square of inclusions on top of it.

$$\begin{array}{ccc} A' \cap B & \xrightarrow{\subseteq} & B \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} \\ A \cup (B \cap A') & \xrightarrow{\subseteq} & A \cup B \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} \\ A' & \xrightarrow[\subseteq]{} & A' \cup B \end{array}$$

Since the outer square $[A' \cap B, A', B, A' \cup B]$ is an intersection-union square, it is a pushout square, thus, by the general properties of pushout squares, in order to show that the square $[A \cup (B \cap A'), A', A \cup B, A' \cup B]$

describes a pushout it is enough to prove that the top square $[A' \cap B, A \cup (B \cap A'), B, A \cup B]$ depicts a pushout. For this we just show that the latter top square is an intersection-union square.

On the one hand we have that

$$(A \cup (B \cap A')) \cup B = A \cup ((B \cap A') \cup B) = A \cup B.$$

On the other hand because $A \subseteq A'$ and $B \cap A' \subseteq A'$ we have $A \cup (B \cap A') \subseteq A'$, thus

$$(A \cup (B \cap A')) \cap B \subseteq A' \cap B. \tag{1}$$

Since $A' \cap B \subseteq B$ and $A' \cap B \subseteq A \cup (B \cap A')$ it follows that

$$A' \cap B \subseteq (A \cup (B \cap A')) \cap B. \tag{2}$$

From (1) and (2) we have that $(A \cup (B \cap A')) \cap B = A' \cap B$. □

Definition 2.9 (Preservation of Objects)**.** In any category endowed with an inclusion system with intersections we say that an arrow $f\colon A \to B$ *preserves* an object C when $(A \cap C \subseteq A)$; f is an inclusion.

Proposition 2.4. *In a category with an epic inclusion system we consider a pushout square as below*

$$\begin{array}{ccc} A & \xrightarrow{\subseteq} & B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ A & \xrightarrow[\subseteq]{} & C \end{array}$$

such that $f; f = f$. Let $f = e_f; (f(A) \subseteq A)$ and $g = e_g; (g(B) \subseteq C)$ with e_f and e_g being abstract surjections.

1. *$f(A) \subseteq g(B)$ and the commutative squares below are pushout squares*

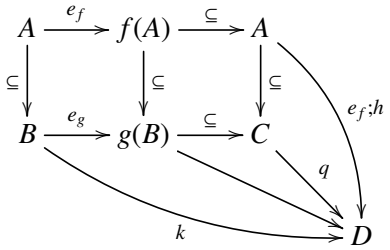
$$\begin{array}{ccccc} A & \xrightarrow{e_f} & f(A) & \xrightarrow{\subseteq} & A \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} & & \downarrow{\scriptstyle\subseteq} \\ B & \xrightarrow[e_g]{} & g(B) & \xrightarrow[\subseteq]{} & C \end{array}$$

2. *Let us in addition assume that the inclusion system has intersections. If g preserves all objects preserved by f then e_g preserves all objects preserved by e_f.*

Proof.
1. By the Diagonal Fill-in Lemma (see [14, 12]) there exists an arrow $f(A) \to g(B)$ which splits the given pushout square into two commutative squares. By the uniqueness of factorization it is immediate to establish that this arrow is an inclusion.

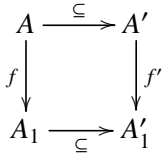
By the general properties of glueing pushout squares together, in order to establish that both squares resulting from this splitting are pushout squares it is enough to establish the pushout property only for

the left-hand side square $[A, B, f(A), g(B)]$. For this we consider $h\colon f(A) \to D$ and $k\colon B \to D$ such that $e_f; h = (A \subseteq B); k$.



$f; f = f$ means $e_f; (f(A) \subseteq A); e_f; (f(A) \subseteq A) = e_f; (f(A) \subseteq A) = e_f; 1_{f(A)}; (f(A) \subseteq A)$. Since each abstract inclusion is mono (see [14, 12]) and since by hypothesis each abstract surjection is epi, we deduce that $(f(A) \subseteq A); e_f = 1_{f(A)}$, hence $(A \subseteq B); k = e_f; h = e_f; (f(A) \subseteq A); e_f; h$. By the pushout property of the outer original square $[A, B, A, C]$ there exists an unique $q\colon C \to D$ such that $g; q = k$ and $(A \subseteq C); q = e_f; h$. We have that $e_g; ((g(B) \subseteq C); q) = k$ and that $(f(A) \subseteq g(B)); ((g(B) \subseteq C); q) = (f(A) \subseteq A); (A \subseteq C); q = (f(A) \subseteq A); e_f; h = h$. The uniqueness of $(g(B) \subseteq C); q$ is given by the epi property of e_g.
2. Let D be an object preserved by e_f, i.e. $(A \cap D \subseteq A); e_f$ is an inclusion. It follows that $(A \cap D \subseteq A); e_f; i_f$ is an inclusion, and thus, by hypothesis, $(B \cap D \subseteq B); e_g; i_g$ is an inclusion too. By the uniqueness of factorization property of inclusion systems we obtain that $(B \cap D \subseteq B); e_g$ is inclusion; therefore D is preserved by e_g. □

Definition 2.10 (Free Extensions along Inclusions)**.** In any category endowed with an inclusion system with intersections we say that an arrow $f\colon A \to A_1$ *admits free extensions along an inclusion* $A \subseteq A'$ when there exist pushout squares of the form



such that every object preserved by f is also preserved by f'.

We say that f *strongly admits free extensions along* $A \subseteq A'$ when for every object A_0 the arrow f admits a free extension f' as above such that $A_0 \cap A'_1 \subseteq A_0 \cap A'$.

Fact 2.2. *In $\mathbb{S}$et (endowed with the standard inclusion system) a function $f\colon A \to A_1$ admits free extensions along any $A \subseteq A'$ if and only if A_1 and $A' \setminus A$ are disjoint. Moreover the free extension $f'\colon A' \to A'_1$ is defined by $A'_1 = A_1 \cup (A' \setminus A)$ and by*

$$f'(a) = \begin{cases} f(a) & \text{when } a \in A, \\ a & \text{otherwise.} \end{cases}$$

Consequently, any function $f\colon A \to A$ strongly admits free extensions along any $A \subseteq A'$.

Definition 2.11 (Idempotent-by-Extension)**.** In a category with pullbacks and endowed with an epic inclusion system, an arrow $f\colon A \to A$ is called *idempotent-by-extension* when it is idempotent, i.e. $f; f = f$, and there exists an object B such that $A = B \cup f(A)$ and B and $f(A)$ are disjoint.

Example 2.4. In $\mathbb{S}et$ (considered with the standard inclusion system) each idempotent function $f\colon A \to A$ is also idempotent-by-extension by taking $B = A \setminus f(A)$.

The category of the **MSA** signatures does not enjoy the identity between idempotency and idempotency-by-extension, as shown by the following example. Let Σ be a signature with one sort and two constants a and b and $\varphi\colon \Sigma \to \Sigma$ that maps both constants to a. Then φ is idempotent but it is not idempotent-by-extension.

3. Structured specifications

This section is structured as follows:

1. We define the concept of structured specification and the corresponding model-oriented denotations; this includes the introduction of our new specification building operators that cover the non-protecting importation situations.
2. We provide several examples of how concrete specification modules can be expressed by our primitive specification building operators.
3. We develop some basic properties of structured specifications.

3.1. Primitive specification building operators

Given an inclusive institution $\mathcal{I} = (\mathbb{S}ig, \mathsf{Sen}, \mathsf{Mod}, \models)$, its *structured specifications* (or just *specifications* for short) are defined from the finite presentations by iteration of several specification building operators. The semantics of each specification SP is given by its *signature* $Sig[SP]$ and its category of *models* $Mod[SP]$. Below we sometimes define only the class of objects for each $Mod[SP]$, the category $Mod[SP]$ being the corresponding full subcategory of $\mathsf{Mod}(Sig[SP])$. For any specification SP we also calculate the *set of its axioms* $Ax[SP] \subseteq \mathsf{Sen}(Sig[SP])$.

Let us fix two classes of signature morphisms $\mathcal{T}, \mathcal{D} \subseteq \mathbb{S}ig$, considered as parameters for the structuring process.

PRES. Each finite presentation (Σ, E) is a specification such that

$$\begin{aligned} Sig[(\Sigma, E)] &= \Sigma, \\ Ax[(\Sigma, E)] &= E, \\ Mod[(\Sigma, E)] &= \mathsf{Mod}(\Sigma, E). \end{aligned}$$

UNION. For any specifications SP_1 and SP_2 we can take their *union* $SP_1 \cup SP_2$ with

$$\begin{aligned} Sig[SP_1 \cup SP_2] &= Sig[SP_1] \cup Sig[SP_2], \\ Ax[SP_1 \cup SP_2] &= Ax[SP_1] \cup Ax[SP_2], \text{[3]} \\ |Mod[SP_1 \cup SP_2]| &= \{M \in \mathsf{Mod}(Sig[SP_1 \cup SP_2]) \mid M\restriction_{Sig[SP_i]} \in Mod[SP_i] \text{ for each } i \in \{1,2\}\}. \end{aligned}$$

TRANS. For any specification SP and signature morphism $(\varphi\colon Sig[SP] \to \Sigma') \in \mathcal{T}$ we can take its *translation along* φ denoted by $SP \star \varphi$ and such that

$$\begin{aligned} Sig[SP \star \varphi] &= \Sigma', \\ Ax[SP \star \varphi] &= \varphi(Ax[SP]), \\ |Mod[SP \star \varphi]| &= \{M' \in \mathsf{Mod}(\Sigma') \mid M'\restriction_{\varphi} \in Mod[SP]\}. \end{aligned}$$

When φ is inclusion we may denote $SP \star \varphi$ by $SP \star \Sigma'$.

DERIV. For any specification SP' and signature morphism $(\varphi\colon \Sigma \to Sig[SP']) \in \mathcal{D}$ we can take its *derivation along* φ denoted by $\varphi \mid SP'$ such that

$$\begin{aligned} Sig[\varphi \mid SP'] &= \Sigma, \\ Ax[\varphi \mid SP'] &= \varphi^{-1}(Ax[SP']^{**}), \\ |Mod[\varphi \mid SP']| &= \{M'\restriction_{\varphi} \mid M' \in Mod[SP']\}. \end{aligned}$$

When φ is inclusion we may denote $\varphi \mid SP'$ by $\Sigma \mid SP'$.

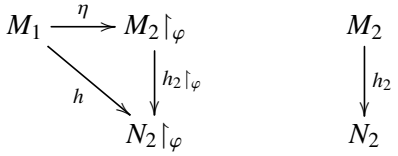
$\mathcal{H}$-EXT. Given a class $\mathcal{H}$ of model homomorphisms, we consider the *$\mathcal{H}$-extension* of a specification SP, denoted $\mathcal{H}(SP)$, such that

$$\begin{aligned} Sig[\mathcal{H}(SP)] &= Sig[SP], \\ Ax[\mathcal{H}(SP)] &= Ax[SP], \text{ and} \\ |Mod[\mathcal{H}(SP)]| &= \{M' \in \mathsf{Mod}(Sig[SP]) \mid M' \models Ax[SP] \text{ and} \\ &\qquad \text{there exists } (h\colon M \to M') \in \mathcal{H} \text{ with } M \in Mod[SP]\} \end{aligned}$$

$\mathcal{H}$-FREE. Given a class $\mathcal{H}$ of model homomorphisms, for any specifications SP_1 and SP_2 and signature morphism $\varphi\colon Sig[SP_1] \to Sig[SP_2]$, we consider the *$\mathcal{H}$-free restriction of SP_2 modulo φ and SP_1*, denoted $SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$, such that

$$\begin{aligned} Sig[SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)] &= Sig[SP_2], \\ Ax[SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)] &= Ax[SP_2], \text{ and} \\ |Mod[SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)]| &= \{M_2 \in Mod[SP_2] \mid \text{there exists } M_1 \in Mod[SP_1] \text{ and} \\ &\qquad \text{an universal arrow } (\eta\colon M_1 \to M_2\restriction_{\varphi}) \in \mathcal{H} \\ &\qquad \text{to the reduct functor } Mod[SP_2] \to \mathsf{Mod}(Sig[SP_1])\}. \end{aligned}$$

This means that for each homomorphism $h\colon M_1 \to N_2\restriction_{\varphi}$ where $N_2 \in Mod[SP_2]$ there exists an unique homomorphism $h_2\colon M_2 \to N_2$ such that $h = \eta; h_2\restriction_{\varphi}$.



When φ is an inclusion of signatures we may omit φ from the notations and denote $SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$ simply by $SP_2\ !_{\mathcal{H}}\ SP_1$. When SP_1 is a presentation of the form $(\Sigma, \emptyset)$, with Σ signature, we may simply write it as Σ and denote the specification $SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$ by $SP_2\ !_{\mathcal{H}}\ \varphi$ or $SP_2\ !_{\mathcal{H}}\ \Sigma$ when φ is inclusion. When $\mathcal{H}$ is the class of identities we omit it as the subscript of !, and the universal property of the models of $SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$ is called *strongly persistently φ-free*.

Remark 3.1.

1. In some of the literature, e.g. [6, 27], the union $\cup$ is usually partially defined, only for specifications over the same signature. The general union of two specifications is then obtained as the (partially defined) union of their translations to the union signature. Like in [14] our use of inclusion systems allows for the direct definition of the union of *any* specifications, without any conditions.
2. Note that if $\mathcal{T}$ and $\mathcal{D}$, resp., are the class of the identities, then TRANS and DERIV, resp. are cancelled. The rather realistic idea to define TRANS and DERIV relative to sub-classes of signature morphisms seems to belong to [6]. Often in practice $\mathcal{D}$ is the class of signature inclusions while $\mathcal{T}$ is the class of all signature morphisms.

3. $\mathcal{H}$-EXT is a completely new operator introduced for capturing non-protecting importation modes.
4. Our operator $\mathcal{H}$-FREE constitutes a significant extension of the existing initial semantics operator that can be found in the literature (such as in [27]) which corresponds to the case when $\mathcal{H}$ is the class of the identities and SP_1 is empty. The extension to arbitrary $\mathcal{H}$ is motivated by the capture of initial semantics in relation with non-protecting importation modes.

3.2. Examples

Example 3.1. The following is a specification of the class of all fields in the CafeOBJ language [13]. The underlying institution of this specification is **OSA**.

```
mod* GROUPS {
  [ G ]
  op 1 :  -> G
  op _*_ : G G -> G {assoc}
  op _-1 : G -> G
  vars x y : G
  eq x * 1 = x .
  eq 1 * x = x .
  eq x * (x -1) = 1 .
  eq (x -1) * x = 1 .
}
mod! GROUPSZ {
  protecting(GROUPS)
  [ G < F ]
  op 0 :  -> F
  op _*_ : F F -> F
  var x : F
  eq x * 0 = 0 .
  eq 0 * x = 0 .
}
mod* FIELDS {
  protecting(GROUPSZ)
  op _+_ : F F -> F {assoc comm}
  op -_ : F -> F
  vars x y z : F
  eq x + 0 = x .
  eq x + (- x) = 0 .
  eq x * (y + z) = (x * y) + (x * z) .
  eq (y + z) * x = (y * x) + (z * x) .
}
```

In this specification GROUPSZ imports GROUPS and FIELDS imports GROUPSZ. The specification GROUPS is flat, its denotation consisting of the class of all groups (with multiplicative notation). Then

$$\texttt{GROUPSZ} = (\texttt{GROUPS} \cup (\Sigma', E')) \,!\, Sig[\texttt{GROUPS}]$$

where Σ' is the extension of Sig[GROUPS] with F (declared as a super-sort of G), 0 and _*_: F F -> F, and E' is the set of the two Σ'-equations introduced by GROUPSZ. The definition of FIELDS is

$$\texttt{FIELDS} = \texttt{GROUPSZ} \cup (\Sigma'', E'')$$

where Σ'' extends Sig[GROUPSZ] with the two operation symbols introduced by FIELDS and E'' consists of the four equations introduced by FIELDS.

Example 3.2. The following **MSA** specification of integer numbers as an 'extension' of the natural numbers uses the CafeOBJ importation mode extending.

```
mod! PNAT {
  [ Number ]
  op 0 :  -> Number
  op s : Number -> Number
}
mod! PINT {
  extending(PNAT)
  op p : Number -> Number
  var X : Number
  eq p(s(X)) = X .
  eq s(p(X)) = X .
}
```

Let EX be the class of **MSA** inclusive model homomorphisms, i.e. model homomorphisms with all the components set theoretic inclusions. Then we have that

$$\texttt{PNAT} = (\Sigma, \emptyset) \,!\, \emptyset$$

where $\Sigma = Sig$[PNAT], the empty signature is also denoted by $\emptyset$ and

$$\texttt{PINT} = (EX(\texttt{PNAT}) \cup (\Sigma', E')) \,!_{EX}\, \texttt{PNAT}$$

where Σ' is the extension of the Sig[PNAT] with the operation symbol p and E' is the set that consists of the two equations introduced by PINT.

One may note that the denotations of tight CafeOBJ modules with non-protecting imports is given by expressions using both $\mathcal{H}$-EXT and $\mathcal{H}$-FREE operators, based upon the same class $\mathcal{H}$ of model homomorphisms. In this example $!_{EX}$ selects those models of $(EX(\texttt{PNAT}) \cup (\Sigma', E'))$ whose reducts to Sig[PNAT] are codomains of universal arrows (belonging to EX) from models of PNAT to the reduct functor from $Mod[EX(\texttt{PNAT}) \cup (\Sigma', E')]$ to Mod(Sig[PNAT]). Consequently, a more restrictive choice than EX as parameter for $\mathcal{H}$-EXT could eliminate the intended models of $EX(\texttt{PNAT}) \cup (\Sigma', E')$, whereas a less restrictive one could impose unnecessary conditions on the models of PINT.

Note that if we used protecting(PNAT) instead of the extending importation mode, then PINT would have been inconsistent in the sense of lacking models. Of course this could have been repaired by introducing a super-sort for the integers, but the cost here would be to involve a more sophisticated logic, namely order sorted algebra. In fact, it is often the case that extending importation modes can be specified alternatively by protecting modes but within order sorted algebra, and in addition to that, one would also have to specify some overloading of function symbols to the new super-sort.

Example 3.3. The following CafeOBJ code represents a **MSA** specification of $\{\mathbb{Z}_n \mid n \in \omega\}$, i.e. the class of the natural numbers modulo n for all $n \in \omega$. The only operations considered are 0 and successor (s).

```
mod* PNATn {
  protecting(PNAT)
  op n : -> PNat
}
mod! Zn {
  using(PNATn)
  eq n = 0 .
}
```

Let *US* be the class of *all* **MSA** model homomorphisms. Then

$$\texttt{PNATn} = \texttt{PNAT} \cup (\Sigma', \emptyset)$$

where Σ' adds the operation n to $Sig[\texttt{PNAT}]$ and

$$\texttt{Zn} = (US\,(\texttt{PNATn}) \cup (\Sigma', \{\texttt{n=0}\}))\ !_{US}\ \texttt{PNATn}.$$

Note that the models of PNATn are the pointed sets of the natural numbers, with the base-point denoted by n. By definitions Zn specifies the free models along the theory inclusion $(\Sigma', \emptyset^{**}) \subseteq (\Sigma', \{\texttt{n=0}\}^{**})$ that are based upon, or generated by, the PNATn models. These are obtained by identifying the elements of any given PNATn algebra according to the congruence modulo n.

3.3. Basic properties of structured specifications

Proposition 3.1. *For each specification SP, $M \in Mod[SP]$ implies $M \models Ax[SP]$.*

Proof. We show the conclusion of the proposition by induction on the structure of the specification *SP*.

$SP = (\Sigma, E)$**:** Obvious from the definition.

$SP = SP_1 \cup SP_2$**:** Let $M \in Mod[SP_1 \cup SP_2]$. Let us denote $Sig[SP_1]$ by Σ_1 and $Sig[SP_2]$ by Σ_2. Since $M|_{\Sigma_k} \in Mod[SP_k]$ for each $k \in \{1, 2\}$, by the induction hypothesis we have that each $M|_{\Sigma_k} \models Ax[SP_k]$. By the satisfaction condition it follows that $M \models_{\Sigma_1 \cup \Sigma_2} Ax[SP_k]$ for each $k \in \{1, 2\}$. Hence $M \models Ax[SP_1] \cup Ax[SP_2]$.

$SP = SP' \star \varphi$**:** Let $M \in Mod[SP' \star \varphi]$. Then $M|_\varphi \in Mod[SP']$ which by the induction hypothesis implies $M|_\varphi \models Ax[SP']$. By the satisfaction condition this implies $M \models \varphi(Ax[SP']) = Ax[SP' \star \varphi]$.

$SP = \varphi \mid SP'$**:** Let $M \in Mod[\varphi \mid SP']$. Then there exists $M' \in Mod[SP']$ such that $M = M'|_\varphi$. By the induction hypothesis we have that $M' \models Ax[SP']$ hence $M' \models Ax[SP']^{**}$ too. But $\rho \in \varphi^{-1}(Ax[SP']^{**})$ means $\varphi(\rho) \in Ax[SP']^{**}$. Hence $M' \models \varphi(\rho)$ which by the satisfaction condition implies that $M \models \rho$. This shows that $M \models \varphi^{-1}(Ax[SP']^{**}) = Ax[\varphi \mid SP']$.

$SP = \mathcal{H}(SP')$**:** Obvious by definition, from the induction hypothesis.

$SP = SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$**:** Obvious by definition, from the induction hypothesis.

□

Following a similar argument it can be shown that the converse of Proposition 3.1 holds for the first three operators only.

Fact 3.1. *For any specification SP built only with* PRES, UNION *and* TRANS *we have*

$$Mod[SP] = \mathsf{Mod}(Sig[SP], Ax[SP]).$$

For the case of the last three operators we give the following counter-arguments:

- In the sub-institution of **MSA** obtained by restricting the sentences only to universally quantified equations, the class of models of a specification $\varphi \mid SP'$ is not necessarily closed under submodels, therefore, in general, it cannot be specified through presentations.
- The same remark as above holds for specifications $\mathcal{H}(SP)$ where $\mathcal{H}$ is the class of strictly inclusive homomorphisms.
- For the last operator let us consider the specification $(\Sigma, E)\ !\ \emptyset$ describing the class of initial models of (Σ, E). Since this class is not closed under products it follows that it cannot be the class of models of a theory.

Fact 3.2. *The following defines a preorder on specifications*

$$SP_1 \models SP_2 \quad \textit{if and only if} \quad Sig[SP_1] = Sig[SP_2] \ \textit{ and } \ Mod[SP_1] \subseteq Mod[SP_2].$$

The Definitions 3.1 and 3.2 together with the Facts 3.3 and 3.4 below can be found in the literature, for example in [27].

Definition 3.1 (Equivalent Specifications)**.** Two specifications SP_1 and SP_2 are *equivalent*, denoted $SP_1 \models\!\!\models SP_2$, when $SP_1 \models SP_2$ and $SP_2 \models SP_1$.

In general it is possible to have different specifications that are equivalent. When we are interested only in the semantics of specifications rather than in the way they are constructed, it does make sense to consider specifications modulo this equivalence relation.

Definition 3.2 (Specification morphisms)**.** A *specification Morphism* $\varphi\colon SP_1 \to SP_2$ between specifications SP_1 and SP_2 is a signature morphism $\varphi\colon Sig[SP_1] \to Sig[SP_2]$ such that $SP_2 \models SP_1 \star \varphi$.

Fact 3.3. *A signature morphism* $\varphi\colon Sig[SP_1] \to Sig[SP_2]$ *is a specification morphism* $SP_1 \to SP_2$ *if and only if* $\varphi \mid SP_2 \models SP_1$.

Fact 3.4. *For any institution* $\mathcal{I}$*, the specifications and their morphisms under the obvious composition form a category, denoted* $\mathbb{S}pec^{\mathcal{I}}$.

The following gives a characterization of isomorphisms of specifications that is useful within the context of the result of Theorem 5.1.

Proposition 3.2. $\varphi\colon SP_1 \to SP_2$ *is an isomorphism of specifications if and only if* $\varphi\colon Sig[SP_1] \to Sig[SP_2]$ *is an isomorphism of signatures and* $SP_1 \star \varphi \models\!\!\models SP_2$.

Proof. For the implication from the left to the right it is immediate that $\varphi\colon Sig[SP_1] \to Sig[SP_2]$ is an isomorphism of signatures. We need only to show that $SP_1 \star \varphi \models SP_2$. For this we consider the inverse φ^{-1}. Since $\varphi^{-1}\colon SP_2 \to SP_1$ is a specification morphism we have that $SP_1 \models SP_2 \star \varphi^{-1}$. Since $\star$ is monotone

with respect to $\models$ we further obtain that $SP_1 \star \varphi \models SP_2 \star \varphi^{-1} \star \varphi$. Now we have just to apply (6) and (7) from below to see that $SP_2 \star \varphi^{-1} \star \varphi \between SP_2$.

For the implication from the right to the left we know that $\varphi\colon Sig[SP_1] \to Sig[SP_2]$ is an isomorphism of signatures and that $SP_1 \star \varphi \between SP_2$. Let φ^{-1} be the inverse of φ as a signature morphism. We have to establish that $SP_1 \models SP_2 \star \varphi^{-1}$. This is achieved by applying $\star\varphi^{-1}$ to both sides of the relation $SP_1 \star \varphi \models SP_2$, by the monotonicity of $\star$ and by (6) and (7) below. □

The following result from [27] extends the famous lifting result of co-limits from signatures to theories from [16]. We recall it together with its proof because later in the paper we will need the explicit construction of pushouts of specification morphisms.

Proposition 3.3. *The forgetful functor $\mathbb{S}pec \to \mathbb{S}ig$ lifts finite co-limits.*

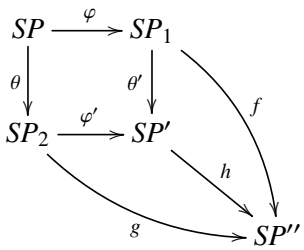
Proof. We use the basic category theory result (see [1]) that each finite co-limit can be expressed in terms of initial objects and pushouts.

For the case of initial objects, it is easy to see that if Σ is an initial signature then $(\Sigma, \emptyset)$ is an initial specification.

For the case of pushouts, we consider any span of specification morphisms $\varphi\colon SP \to SP_1$ and $\theta\colon SP \to SP_2$ and we take a pushout of the underlying signature morphisms as follows.

$$\begin{array}{ccc} Sig[SP] & \xrightarrow{\varphi} & Sig[SP_1] \\ {\scriptstyle\theta}\downarrow & & \downarrow{\scriptstyle\theta'} \\ Sig[SP_2] & \xrightarrow[\varphi']{} & \Sigma' \end{array}$$

We define the specification $SP' = SP_1 \star \theta' \cup SP_2 \star \varphi'$. It is easy to see that $\theta'\colon SP_1 \to SP'$ and $\varphi'\colon SP_2 \to SP'$ are specification morphisms.



For any specification morphisms $f\colon SP_1 \to SP''$ and $g\colon SP_2 \to SP''$ such that $\varphi; f = \theta; g$, by the pushout property for the underlying signature morphisms, there exists an unique signature morphism $h\colon Sig[SP'] \to \Sigma = Sig[SP'']$ such that $f = \theta'; h$ and $g = \varphi'; h$. It remains to show that h is a specification morphism $SP' \to SP''$. By (6) of Fact 4.1 and (10) of Proposition 4.2 we have that

$$SP' \star h = (SP_1 \star \theta' \cup SP_2 \star \varphi') \star h = SP_1 \star (\theta'; h) \cup SP_2 \star (\varphi'; h) = SP_1 \star f \cup SP_2 \star g.$$

Because f and g are specification morphisms $SP'' \models SP_1 \star f$ and $SP'' \models SP_2 \star g$, hence $SP'' \models SP_1 \star f \cup SP_2 \star g = SP' \star h$. □

Note co-limits of signatures are not lifted uniquely to co-limits of specifications. One argument for this is that for any fixed initial signature Σ we have that any specification (Σ, E) such that $E \subseteq \emptyset^{**}$ is initial.

4. Algebraic rules for structured specifications

In this section we first recall from the algebraic specification folklore and literature some important algebraic rules for the model oriented denotations of structured specifications and after that we prove a series of new rules.

The proofs of Proposition 4.1, Fact 4.1 and of Proposition 4.2 below are straightforward and moreover these results appear elsewhere in the literature (modulo our use of inclusion systems), such as in [27]. In their property oriented variant they can also be found in [14]. Therefore let us skip their proof here.

Proposition 4.1. *For any specifications SP, SP', SP'',*

$$SP \cup SP' \between SP' \cup SP. \tag{3}$$

$$SP \cup SP \between SP. \tag{4}$$

$$(SP \cup SP') \cup SP'' \between SP \cup (SP' \cup SP''). \tag{5}$$

Fact 4.1. *For any signature morphisms $\varphi\colon \Sigma \to \Sigma'$ and $\varphi'\colon \Sigma' \to \Sigma''$ and specifications SP and SP'' such that $Sig[SP] = \Sigma$ and $Sig[SP''] = \Sigma''$,*

$$SP \star (\varphi;\varphi') \between (SP \star \varphi) \star \varphi'. \tag{6}$$

$$SP \star 1_\Sigma \between SP. \tag{7}$$

$$(\varphi;\varphi') \mid SP'' \between \varphi \mid \varphi' \mid SP''. \tag{8}$$

$$1_{\Sigma'} \mid SP' \between SP'. \tag{9}$$

Proposition 4.2. *For any specifications SP_1 and SP_2 and any signature morphism $\varphi\colon Sig[SP_1 \cup SP_2] \to \Sigma$*

$$(SP_1 \cup SP_2) \star \varphi \between (SP_1 \star (i_1;\varphi)) \cup (SP_2 \star (i_2;\varphi)) \tag{10}$$

where i_k is the inclusion $Sig[SP_k] \subseteq Sig[SP_1 \cup SP_2]$ for $k \in \{1,2\}$.

The following has been proved in [27].

Proposition 4.3. *For any pushout of signature morphisms as below*

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi} & \Sigma_1 \\ {\scriptstyle\theta}\downarrow & & \downarrow{\scriptstyle\theta'} \\ \Sigma_2 & \xrightarrow[\varphi']{} & \Sigma' \end{array}$$

and for any specification SP_1 with $Sig[SP_1] = \Sigma_1$

$$\varphi' \mid (SP_1 \star \theta') \models (\varphi \mid SP_1) \star \theta. \tag{11}$$

If the institution has weak model amalgamation then

$$\varphi' \mid (SP_1 \star \theta') \between (\varphi \mid SP_1) \star \theta. \tag{12}$$

Proposition 4.4. *In any institution, for any pushout of signatures as below*

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\theta_1} \\ \Sigma_2 & \xrightarrow[\theta_2]{} & \Sigma' \end{array}$$

and for any specifications SP_1, SP_2 such that $\Sigma_k = Sig[SP_k]$ for $k \in \{1,2\}$ we have that

$$(\varphi_k;\theta_k) \mid (SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2) \models (\varphi_1 \mid SP_1) \cup (\varphi_2 \mid SP_2), \ \ \textit{for } k \in \{1,2\}. \tag{13}$$

If the institution has weak model amalgamation then

$$(\varphi_k;\theta_k) \mid (SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2) \models\!\!\models (\varphi_1 \mid SP_1) \cup (\varphi_2 \mid SP_2), \ \ \textit{for } k \in \{1,2\}. \tag{14}$$

Proof.
(13): Let $M \in Mod[(\varphi_k;\theta_k) \mid (SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2)]$. Then there exists $M' \in Mod[SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2]$ such that $M = M'\restriction_{\varphi_k;\theta_k}$. Let $M_1 = M'\restriction_{\theta_1}$ and $M_2 = M'\restriction_{\theta_2}$. Then $M_i \in Mod[SP_i]$ for $i \in \{1,2\}$. Since $M = M_i\restriction_{\varphi_i}$ for $i \in \{1,2\}$ we have that $M \in Mod[\varphi_i \mid SP_i]$ for $i \in \{1,2\}$ which means $M \in Mod[\varphi_1 \mid SP_1] \cap Mod[\varphi_2 \mid SP_2] = Mod[(\varphi_1 \mid SP_1) \cup (\varphi_2 \mid SP_2)]$.
(14): Let $M \in Mod[(\varphi_1 \mid SP_1) \cup (\varphi_2 \mid SP_2)]$. Since $Mod[(\varphi_1 \mid SP_1) \cup (\varphi_2 \mid SP_2)] = Mod[\varphi_1 \mid SP_1] \cap Mod[\varphi_2 \mid SP_2]$ for each $i \in \{1,2\}$ there exists $M_i \in Mod[SP_i]$ such that $M = M_i\restriction_{\varphi_i}$. By the weak model amalgamation hypothesis there exists a Σ'-model M' such that $M'\restriction_{\theta_i} = M_i$. This means $M' \in Mod[SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2]$. Since $M = M'\restriction_{\varphi_k;\theta_k}$ we have that $M \in Mod[(\varphi_k;\theta_k) \mid (SP_1 \star \theta_1 \ \cup \ SP_2 \star \theta_2)]$. □

Corollary 4.1. *In any institution with unions and intersections of signatures, for any specifications SP_1 and SP_2, let $\Sigma = Sig[SP_1] \cap Sig[SP_2]$. Then*

$$\Sigma \mid (SP_1 \cup SP_2) \models (\Sigma \mid SP_1) \cup (\Sigma \mid SP_2). \tag{15}$$

Moreover if the institution has weak model amalgamation and each intersection-union square of signatures is pushout then

$$\Sigma \mid (SP_1 \cup SP_2) \models\!\!\models (\Sigma \mid SP_1) \cup (\Sigma \mid SP_2). \tag{16}$$

The distributivity rule (16) above has been stated as an exercise in [27] for the particular case of equational logic. Its property oriented variant has been a cornerstone in [3] (for the special case of many sorted first order logic) and in [14] (in the general institution-independent case), its proof has been significantly more difficult that the proof above of its model oriented variant and required an interpolation property for the underlying institution.

Fact 4.2. *If $\mathcal{H}$ contains all identities then for each flat specification (Σ, E)*

$$\mathcal{H}(\Sigma, E) \models\!\!\models (\Sigma, E). \tag{17}$$

Fact 4.3. *If $\mathcal{H} = \mathcal{H}';\mathcal{H}''$ then for each specification SP*

$$\mathcal{H}'(\mathcal{H}''(SP)) \models\!\!\models \mathcal{H}(SP). \tag{18}$$

Definition 4.1. A class $\mathcal{H}$ of model homomorphisms is preserved by a signature morphism φ when $h\restriction_{\varphi} \in \mathcal{H}$ for each $h \in \mathcal{H}$.

Proposition 4.5. *If each inclusion of signatures preserves $\mathcal{H}$ then for all specifications SP_1 and SP_2 we have*

$$\mathcal{H}(SP_1 \cup SP_2) \models \mathcal{H}(SP_1) \cup \mathcal{H}(SP_2) \tag{19}$$

Proof.
(19): Let $M' \in Mod[\mathcal{H}(SP_1 \cup SP_2)]$. Then there exists $N' \in Mod[SP_1 \cup SP_2]$ and $(h' \colon N' \to M') \in \mathcal{H}$ such that $M' \models Ax[SP_1] \cup Ax[SP_2]$. It follows that $h'|_{\Sigma_k} \in \mathcal{H}$ for $k \in \{1, 2\}$. By the satisfaction condition $M'|_{\Sigma_k} \models Ax[SP_k]$. These imply $M' \in Mod[\mathcal{H}(SP_1) \cup \mathcal{H}(SP_2)]$. □

Recall from [12] the following concept:

Definition 4.2 (Lifting of Relations)**.** Let $\varphi \colon \Sigma_1 \to \Sigma_2$ be a signature morphism and $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2 \rangle$ with $\mathcal{R}_1 \subseteq |\mathsf{Mod}(\Sigma_1)| \times |\mathsf{Mod}(\Sigma_1)|$ and $\mathcal{R}_2 \subseteq |\mathsf{Mod}(\Sigma_2)| \times |\mathsf{Mod}(\Sigma_2)|$ be a pair of binary relations. We say that φ *lifts* $\mathcal{R}$ if and only if for each $M_2 \in |\mathsf{Mod}(\Sigma_2)|$ and $N_1 \in |\mathsf{Mod}(\Sigma_1)|$, if $\langle M_2|_\varphi, N_1 \rangle \in \mathcal{R}_1$, there exists $N_2 \in |\mathsf{Mod}(\Sigma_2)|$ such that $N_2|_\varphi = N_1$ and $\langle M_2, N_2 \rangle \in \mathcal{R}_2$.

$$M_2|_\varphi \xrightarrow{\mathcal{R}_1} N_1 = N_2|_\varphi$$
$$M_2 \xrightarrow[\mathcal{R}_2]{} (\exists) N_2$$

Proposition 4.6. *If φ lifts $\xleftarrow{\mathcal{H}}$ then*

$$\mathcal{H}(SP) \star \varphi \models \mathcal{H}(SP \star \varphi). \tag{20}$$

If φ lifts $\xrightarrow{\mathcal{H}}$ and $\mathcal{H}$ preserves the satisfaction of all sentences of the institution then

$$\mathcal{H}(\varphi \mid SP') \models \varphi \mid \mathcal{H}(SP'). \tag{21}$$

If φ preserves $\mathcal{H}$ then

$$\mathcal{H}(SP \star \varphi) \models \mathcal{H}(SP) \star \varphi. \tag{22}$$

$$\varphi \mid \mathcal{H}(SP') \models \mathcal{H}(\varphi \mid SP'). \tag{23}$$

Proof.
(20): Let $N' \in Mod[\mathcal{H}(SP) \star \varphi]$. Then there exists $M \in Mod[SP]$ and $(h \colon M \to N'|_\varphi) \in \mathcal{H}$ such that $N'|_\varphi \models Ax[SP]$. By the lifting assumption there exists $(h' \colon M' \to N') \in \mathcal{H}$ such that $M'|_\varphi = M$. By the Satisfaction Condition we have $M' \models \varphi(Ax[SP]) = Ax[SP \star \varphi]$. Hence $N' \in Mod[\mathcal{H}(SP \star \varphi)]$.
(21): Let $N \in Mod[\mathcal{H}(\varphi \mid SP')]$. Then there exists $M' \in Mod[SP']$ and $(h \colon M'|_\varphi \to N) \in \mathcal{H}$. By the lifting assumption there exists $(h' \colon M' \to N') \in \mathcal{H}$ such that $N'|_\varphi = N$. By the preservation assumption $N' \models Ax[SP']$, hence $N' \in Mod[\mathcal{H}(SP')]$.
(22): Let us consider $N' \in Mod[\mathcal{H}(SP \star \varphi)]$. Then there exists $(h' \colon M' \to N') \in \mathcal{H}$ such that $M'|_\varphi \in Mod[SP]$. Also $N' \models Ax[SP \star \varphi] = \varphi(Ax[SP])$. By the preservation assumption we have that $(h'|_\varphi \colon M'|_\varphi \to N'|_\varphi) \in \mathcal{H}$. By the Satisfaction Condition we have $N'|_\varphi \models Ax[SP]$. Hence $N' \in Mod[\mathcal{H}(SP) \star \varphi]$.
(23): Let $N \in Mod[\varphi \mid \mathcal{H}(SP')]$. Then there exists a model $N' \in Mod[\mathcal{H}(SP')]$ with $N'|_\varphi = N$ and $(h' \colon M' \to N') \in \mathcal{H}$ such that $M' \in Mod[SP']$ and $N' \models Ax[SP']$. By the preservation assumption $(h'|_\varphi \colon M'|_\varphi \to N'|_\varphi) \in \mathcal{H}$. By the Satisfaction Condition we have that $N'|_\varphi \models \varphi^{-1}(Ax[SP']^{**}) = Ax[\varphi \mid SP']$ hence $N'|_\varphi \in Mod[\mathcal{H}(\varphi \mid SP')]$. □

Example 4.1. Let *EX* and *EPI* be the classes of the **MSA** model homomorphisms that are inclusions and surjective, resp. Let *US* be the class of all **MSA** model homomorphisms and *ISO* the class of **MSA** model isomorphisms. We say that an **MSA** signature morphism $\varphi\colon (S, F) \to (S', F')$ is an *encapsulation* if for each $\sigma' \in F'_{w' \to \varphi(s)}$ there exists σ in F such that $\sigma' = \varphi(\sigma)$. According to the literature (e.g. [11, 12, 25] we have the following:

- φ lifts (model) isomorphisms if and only if it is injective on the sorts.
- φ lifts $\overset{EX}{\to}$ and $\overset{EPI}{\leftarrow}$ if it is injective.
- φ lifts $\overset{EPI}{\to}$ and $\overset{EX}{\leftarrow}$ if it is injective on the sorts and it is an encapsulation.
- φ lifts $\overset{US}{\to}$ and $\overset{US}{\leftarrow}$ if it is both injective and an encapsulation.

It is very easy to check the following:

- Any φ preserves *EX*, *EPI*, and *US*.

The following holds by the basic assumption on our institutions:

- *ISO* preserves the satisfaction of all sentences.

The following properties are well known from the model theory literature (e.g. [21]):

- *EX* preserves the satisfaction of the sentences of the form $(\exists X)\rho$ where ρ is any quantifier-free sentence.
- *EPI* preserves the satisfaction of the universally quantified equations $(\forall X)t = t'$.
- *US* preserves the satisfaction of the equational atoms $t = t'$.

Corollary 4.2. *If each morphism in $\mathcal{D}$ (i.e. used for derivation) lifts isomorphisms and $\mathcal{H}; ISO \subseteq ISO; \mathcal{H}$ then the class of models $Mod[SP]$ of each specification SP is closed under isomorphisms.*

Proof. We prove the conclusion of the proposition by recursion on the structure of the specification *SP*.

$SP = (\Sigma, E)$**:** From (17) of Fact 4.2 with *ISO* in the role of $\mathcal{H}$ there.

$SP = SP_1 \cup SP_2$**:** From (19) of Proposition 4.5 with *ISO* in the role of $\mathcal{H}$ there.

$SP = SP' \star \varphi$**:** From (22) of Proposition 4.6 with *ISO* in the role of $\mathcal{H}$ there.

$SP = \varphi \mid SP'$**:** From (21) of Proposition 4.6 with *ISO* in the role of $\mathcal{H}$ there.

$SP = \mathcal{H}(SP')$**:** Let $M \in Mod[SP]$ and $N \cong M$. Note that by the assumption that the institution is closed under isomorphisms, $N \models Ax[SP']$. There exists $M' \in Mod[SP']$ and $(h\colon M' \to M) \in \mathcal{H}$. Because $\mathcal{H}; ISO \subseteq ISO; \mathcal{H}$ there exists $N' \cong M'$ and $(f\colon N' \to N) \in \mathcal{H}$. By the induction hypothesis $N' \in Mod[SP']$ hence $N \in Mod[SP]$.

$SP = SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)$**:** Let $M_2 \in Mod[SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)]$ and let $i_2\colon M_2 \to N_2$ be an isomorphism. This means there exists $M_1 \in Mod[SP_1]$ and $(\eta\colon M_1 \to M_2{\restriction_\varphi}) \in \mathcal{H}$ universal arrow. Because $\mathcal{H}; ISO \subseteq ISO; \mathcal{H}$ there exists N_1, $i_1\colon M_1 \to N_1$ isomorphism, and $(\eta'\colon N_1 \to N_2{\restriction_\varphi}) \in \mathcal{H}$ such that the diagram below commutes.

$$\begin{array}{ccc} M_2{\restriction_\varphi} & \xrightarrow[\cong]{i_2{\restriction_\varphi}} & N_2{\restriction_\varphi} \\ {\scriptstyle\eta}\uparrow & & \uparrow{\scriptstyle\eta'} \\ M_1 & \xrightarrow[i_1]{\cong} & N_1 \end{array}$$

Let $M'_2 \in Mod[SP_2]$ and let homomorphism $h\colon N_1 \to M'_2{\restriction_\varphi}$. By the universal property of (η, M_2) there exists an unique homomorphism $f'\colon M_2 \to M'_2$ such that $\eta; f'{\restriction_\varphi} = i_1; h$. Then $\eta'; (i_2^{-1}; f'){\restriction_\varphi} = i_1^{-1}; \eta; f'{\restriction_\varphi} = i_1^{-1}; i_1; h = h$. This shows the existence part. Moreover for any $g'\colon N_2 \to M'_2$ such that $\eta'; g'{\restriction_\varphi} = h$ we have that $i_1; \eta'; g'{\restriction_\varphi} = \eta; (i_2; g'){\restriction_\varphi}$ and by the uniqueness of f' this implies $i_2; g' = f'$. Hence g' *must be* $i_2^{-1}; f'$ indeed. We have thus shown that there exists an unique $g'\colon N_2 \to M'_2$ such that $\eta'; g'{\restriction_\varphi} = h$ which proves that $N \in Mod[SP_2\ !_{\mathcal{H}}\ (\varphi, SP_1)]$.

□

Example 4.2. For the **MSA** specifications that are structured with *EX* and/or *US* and such that each morphism in $\mathcal{D}$ is injective on the sorts, from Corollary 4.2 we have that $Mod[SP]$ is closed under isomorphisms for each structured specification SP. Note that in this case the condition on $\mathcal{D}$ is rather mild, since in practice the information hiding operator DERIV is usually considered for signature inclusions.

Proposition 4.7. *Assume the institution is semi-exact. For any pushout of signatures as below*

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\theta_1} \\ \Sigma_2 & \xrightarrow[\theta_2]{} & \Sigma' \end{array}$$

and for any specifications SP_1, SP_2 and SP such that $\Sigma_k = Sig[SP_k]$ for $k \in \{1,2\}$ and $\Sigma = Sig[SP]$ we have

$$(SP_1 \star \theta_1\ \cup\ SP_2 \star \theta_2)\ !_{\mathcal{H}}\ ((\varphi_k; \theta_k), SP) \models (SP_1\ !_{\mathcal{H}}\ (\varphi_1, SP)) \star \theta_1\ \cup\ (SP_2\ !_{\mathcal{H}}\ (\varphi_2, SP)) \star \theta_2 \quad (24)$$
$$\text{if}\ \ (\varphi_1 \mid SP_1) \between (\varphi_2 \mid SP_2).$$

Proof. Let $M' \in Mod[(SP_1 \star \theta_1\ \cup\ SP_2 \star \theta_2)\ !_{\mathcal{H}}\ ((\varphi_k; \theta_k), SP)]$. Let us employ the following notations: $M_1 = M'{\restriction_{\theta_1}}$, $M_2 = M'{\restriction_{\theta_2}}$, and $M = M'{\restriction_{\varphi_k;\theta_k}}$. We have to show that $M_k \in Mod[SP_k\ !_{\mathcal{H}}\ (\varphi_k, SP)]$ for each $k \in \{1,2\}$.

By the hypothesis there exists $(\eta\colon M_0 \to M) \in \mathcal{H}$ such that $M_0 \in Mod[SP]$ and for any $N' \in Mod[SP_1 \star \theta_1\ \cup\ SP_2 \star \theta_2]$ and $h\colon M_0 \to N'{\restriction_{\varphi_k;\theta_k}}$ there exists an unique $h'\colon M' \to N'$ such that $\eta; h{\restriction_{\varphi_k;\theta_k}} = h$.

Let us fix $k \in \{1,2\}$. For any homomorphism $h_k\colon M_0 \to N_k{\restriction_{\varphi_k}}$ with $N_k \in Mod[SP_k]$ we show that there exists an unique homomorphism $h'_k\colon M_k \to N_k$ such that $\eta; h'_k{\restriction_{\varphi_k}} = h_k$. For showing this we let $\{j\} = \{1,2\} \setminus \{k\}$. From the condition of our equivalence, i.e. that $\varphi_1 \mid SP_1 \between \varphi_2 \mid SP_2$, there exists $N_j \in Mod[SP_j]$ such that $N_k{\restriction_{\varphi_k}} = N_j{\restriction_{\varphi_j}}$. Let N' be the amalgamation of N_k and N_j; evidently $N' \in Mod[SP_1 \star \theta_1 \cup SP_2 \star \theta_2]$. We let $h'\colon M' \to N'$ be the unique homomorphism such that $\eta; h'{\restriction_{\varphi_k;\theta_k}} = h_k$.

The existence of $h'_k\colon M_k \to N_k$ such that $\eta; h'_k{\restriction_{\varphi_k}} = h_k$ is given by defining $h'_k = h'{\restriction_{\theta_k}}$. For showing the uniqueness of h'_k let us we consider $f_k\colon M_k \to N_k$ such that $\eta; f_k{\restriction_{\varphi_k}} = h_k$. Let f be the amalgamation of f_k and $h'{\restriction_{\theta_j}}$. Since $\eta; f{\restriction_{\varphi_k;\theta_k}} = \eta; f_k{\restriction_{\varphi_k}} = h_k$, by the uniqueness part of the universal property of M' we have that $f = h'$, hence $f_k = h'{\restriction_{\theta_k}} = h'_k$. □

Proposition 4.8. *Assume the institution is semi-exact. For any pushout of signatures as below*

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\theta_1} \\ \Sigma_2 & \xrightarrow[\theta_2]{} & \Sigma' \end{array}$$

and for any specifications SP_1 and SP_2 such that $\Sigma_k = Sig[SP_k]$ for $k \in \{1,2\}$ we have

$$(SP_1 \,!\, \varphi_1) \star \theta_1 \;\cup\; (SP_2 \,!\, \varphi_2) \star \theta_2 \models (SP_1 \star \theta_1 \;\cup\; SP_2 \star \theta_2) \,!\, (\varphi_k; \theta_k) \tag{25}$$

$$(SP_1 \star \theta_1 \;\cup\; SP_2 \star \theta_2) \,!\, (\varphi_k; \theta_k) \between (SP_1 \,!\, \varphi_1) \star \theta_1 \;\cup\; (SP_2 \,!\, \varphi_2) \star \theta_2 \;\; \textit{if} \;\; (\varphi_1 \mid SP_1) \between (\varphi_2 \mid SP_2). \tag{26}$$

Proof.
(25): Let $M' \in Mod[(SP_1 \,!\, \varphi_1) \star \theta_1 \;\cup\; (SP_2 \,!\, \varphi_2) \star \theta_2]$. For each $k \in \{1,2\}$ let M_k denote $M'|_{\theta_k}$ and let M denote $M'|_{\varphi_k;\theta_k}$. We have that each M_k, $k \in \{1,2\}$, is strongly persistently φ_k-free and we show that M' is strongly persistently $(\varphi_k;\theta_k)$-free. For this let us consider any homomorphism $h\colon M \to N'|_{\varphi_k;\theta_k}$ for any $N' \in Mod[SP_1 \star \theta_1 \;\cup\; SP_2 \star \theta_2]$. Since from the hypothesis each M_k is strongly persistently φ_k-free we have that there exists $h_k\colon M_k \to N'|_{\theta_k}$ such that $h_k|_{\varphi_k} = h$. Let $h'\colon M' \to N'$ be the amalgamation of h_1 and h_2. Evidently $h'|_{\varphi_k;\theta_k} = h$.

For the uniqueness part, let us consider another homomorphism $f'\colon M' \to N'$ such that $f'|_{\varphi_k;\theta_k} = h$. This implies that $f'|_{\theta_k}|_{\varphi_k} = h$. By the uniqueness part of the universal properties of each M_k, we have that $f'|_{\theta_k} = h_k$, and from this by the uniqueness of amalgamation of homomorphisms we obtain that $f' = h'$.
(26): From (25) and the instance of (24) of Proposition 4.7 above obtained for the class $\mathcal{H}$ of model homomorphisms consisting of the identities and SP consisting of an empty presentation $(\Sigma, \emptyset)$. □

Corollary 4.3. *In any semi-exact institution in which the intersection-union squares of signatures are pushouts we have that for any specifications SP_1 and SP_2 and for $\Sigma = Sig[SP_1] \cap Sig[SP_2]$*

$$(SP_1 \,!\, \Sigma) \cup (SP_2 \,!\, \Sigma) \models (SP_1 \cup SP_2) \,!\, \Sigma \;\; \textit{and} \tag{27}$$

$$(SP_1 \cup SP_2) \,!\, \Sigma \between (SP_1 \,!\, \Sigma) \cup (SP_2 \,!\, \Sigma) \;\; \textit{if} \;\; (\Sigma \mid SP_1) \between (\Sigma \mid SP_2). \tag{28}$$

Example 4.3. Let

```
mod* TRIV { [ Elt ] }
```

According to Corollary 4.3 we have the equivalence TUPLES ⋈ (PAIRS ∪ TRIPLES) of the specifications below:

```
mod! PAIRS {
  protecting(TRIV)
  [ Pairs ]
  op <_,_> : Elt Elt -> Pairs
  ops p1 p2 : Pairs -> Elt
  vars E1 E2 : Elt
  eq p1(<E1,E2>) = E1 .
  eq p2(<E1,E2>) = E2 .
}
mod! TRIPLES {
  protecting(TRIV)
  [ Triples ]
  op <_,_,_> : Elt Elt Elt -> Triples
  ops p1 p2 p3 : Triples -> Elt
  vars E1 E2 E3 : Elt
  eq p1(<E1,E2,E3>) = E1 .
  eq p2(<E1,E2,E3>) = E2 .
  eq p3(<E1,E2,E3>) = E3 .
}
```

```
mod! TUPLES {
  protecting(TRIV)
  [ Pairs Triples ]
  op <_,_> : Elt Elt -> Pairs
  op <_,_,_> : Elt Elt Elt -> Triples
  ops p1 p2 : Pairs -> Elt
  ops p1 p2 p3 : Triples -> Elt
  vars E1 E2 E3 : Elt
  eq p1(<E1,E2>) = E1 .
  eq p2(<E1,E2>) = E2 .
  eq p1(<E1,E2,E3>) = E1 .
  eq p2(<E1,E2,E3>) = E2 .
  eq p3(<E1,E2,E3>) = E3 .
}
```

Proposition 4.9. *If the institution is semi-exact, then for any pushout of signatures*

$$\begin{array}{ccc} \Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\ {\scriptstyle\varphi_2}\downarrow & & \downarrow{\scriptstyle\theta_1} \\ \Sigma_2 & \xrightarrow[\theta_2]{} & \Sigma' \end{array}$$

and any specification SP' such that $Sig[SP'] = \Sigma'$ we have that

$$(\theta_1 \mid SP') \,!\, \varphi_1 \models \theta_1 \mid (SP' \,!\, \theta_2). \tag{29}$$

Proof.
(29): Let $M_1 \in Mod[(\theta_1 \mid SP') \,!\, \varphi_1]$. Then there exists $M' \in Mod[SP']$ such that $M_1 = M'\restriction_{\theta_1}$. It would be sufficient if we showed that $M' \in Mod[SP' \,!\, \theta_2]$. For this we consider any homomorphism $h_2 \colon M_2 = M'\restriction_{\theta_2} \to N'\restriction_{\theta_2}$ for some $N' \in Mod[SP']$. We have to show that there exists an unique homomorphism $h' \colon M' \to N'$ expanding h_2. Since $M_1 \in Mod[(\theta_1 \mid SP') \,!\, \varphi_1]$ there exists an unique homomorphism $h_1 \colon M_1 \to N'\restriction_{\theta_1}$ expanding $h_2\restriction_{\varphi_2}$. By semi-exactness we define $h' \colon M' \to N'$ as $h_1 \otimes h_2$, i.e. the (unique) amalgamation of h_1 and h_2. The uniqueness of h' is a consequence of the uniqueness of the amalgamation and of the uniqueness of the expansion of $h_2\restriction_{\varphi_2}$ to h_1. □

5. On parameterized specification

Pushout-style parameterization originates from work on Clear [7] and constitutes the basis of parameterized specification for the whole OBJ family of languages (i.e. OBJ3 [20], CafeOBJ [13], etc.) but also for ACT TWO [15] and other languages. In this section we develop an institution-independent semantics for pushout-style parameterization that refines the existing one by considering the possible sharing between the body of the parameterized module and the instance of the parameter. It is quite straightforward to note that this consideration fits most realistically and pragmatically the actual practice of parameterized specification and programming; in this section we provide some simple and natural examples supporting this claim. Our approach to (pushout-style) parameterization owes crucially to the use of inclusion systems.

This section is structured as follows:

1. We define the concept of parameterized module and of pushout-style instantiation of parameters.
2. We discuss multiple parameters and their simultaneous (parallel) instantiation as a special case of single parameter instantiation.
3. We introduce sequential (serial) instantiation of multiple parameters and prove a theorem giving a set of sufficient abstract conditions for the isomorphism between the simultaneous and the sequential instantiation of parameters.

5.1. Single parameters

Definition 5.1 (Parameterized Specification)**.** A *parameterized specification*, denoted $SP(P)$, consists of a specification morphism $P \to SP$ such that its underlying signature morphism is an inclusion $Sig[P] \subseteq Sig[SP]$. Then P is called the *parameter* of the parameterized specification and SP the body of the parameterized specification.

In practice, the parameter P is an (isomorphic) renaming of a specification P_0 such that $Sig[P_0]$ and $Sig[P]$ are disjoint. If we denote by p the corresponding isomorphism $Sig[P_0] \to Sig[P]$, then of course $P = P_0 \star p$. The readers familiar with the OBJ family of languages may find that our $SP(P)$ corresponds there to the $SP(p :: P_0)$. The reason for such isomorphic renamings is that while usually we specify P_0, we also need to make sure the parameter does not share with other parts of the our specifications, such as other parameters or specifications used for instantiations. A practical way to achieve this, which is realized in some implementations of actual specification languages, is to rename the entities of P_0 by qualifying them by P. For example a sort s of P_0 would appear in P as $s.P$.

In the literature (e.g. [27]) parameterized specifications are often defined just as specification morphisms $P \to SP$. We think that this is much too general and does not capture precisely enough the realities of parameterized specifications, our additional condition that $Sig[P] \subseteq Sig[SP]$ filling this conceptual gap. Below we will see that one of the consequences of our inclusion systems based approach is the possibility to consider sharing in a rather natural and clean way.

Example 5.1. In the following parameterized specification `SG^` of semigroups 'with powers', the parameter consists of the the renaming of the specification `SG` of semigroups by `S`. In the CafeOBJ notation this is denoted `(S :: SG)`.

```
mod* SG {
  [ Elt ]
  op _+_ : Elt Elt -> Elt { assoc }
}
mod! PNAT {
  [ PNat ]
  op 0 : -> PNat
  op s_ : PNat -> PNat
}
mod! PNAT+ {
  protecting(PNAT)
  op _+_ : PNat PNat -> PNat
  vars M N : PNat
  eq M + 0 = M .
  eq M + s(N) = s(M + N) .
}
mod* SG^ (S :: SG) {
  protecting(PNAT)
  op _^_ : Elt PNat -> Elt
  eq E:Elt ^ s(N:PNat) = E + (E ^ N) .
}
```

In the parameterized specification `SG^`, the sort of `SG` $\star$ `S` is `Elt.S`. In this example the specification `SG^` is defined as (`SG` $\star$ `S`) $\cup$ `PNAT` $\cup$ (Σ', E') where Σ' is Sig[`SG` $\star$ `S`] $\cup$ Sig[`PNAT`] plus the operation `_^_` and E' consists of the only equation specified by `SG^`.

Definition 5.2 (Instantiation of Parameters)**.** Let us consider a parameterized specification $SP(P)$. Given any specification morphism $v\colon P \to SP_1$ such that $Sig[P]$ and $Sig[SP_1]$ are disjoint, *the instance of the*

parameterized specification $SP(P)$ *by* v, denoted $SP(P \Leftarrow v)$, is defined as a pushout of specifications as below

$$\begin{array}{ccc} (Sig[P] \cup (Sig[SP] \cap Sig[SP_1]), \emptyset) & \xrightarrow{\subseteq} & SP \\ {\scriptstyle v+\mathrm{id}}\downarrow & & \downarrow{\scriptstyle v_1} \\ SP_1 & \xrightarrow[i]{} & SP(P \Leftarrow v) \end{array}$$

where

- id is the inclusion $Sig[SP] \cap Sig[SP_1] \subseteq Sig[SP_1]$, and
- v + id is the unique signature morphism 'extending' both v and id by the coproduct property of the disjoint union $Sig[P] \cup (Sig[SP] \cap Sig[SP_1])$ (see Corollary 2.1 and Proposition 2.1).

Note that the instances of parameterized specifications are unique only up to isomorphisms. Also the pushout above takes into account the possible sharing between the body SP of the parameterized module and the instance SP_1 of the parameter. In practice, since the parameters are qualified by renamings (e.g. $\mathtt{SG} \star \mathtt{S}$) the condition that $Sig[P]$ and $Sig[SP_1]$ are disjoint is naturally fulfilled, which means there is no need to consider the more general case with sharing between P and SP_1 that may lead to technical complications (such as conditions on v).

The lifting co-limit result of Proposition 3.3 gives the following two-steps characterization for instances of parameterized specifications.

Corollary 5.1 (Instantiation of Parameters)**.** *$SP(P \Leftarrow v)$ of Definition 5.2 may be obtained as follows:*

1. *We consider a pushout square of signature morphisms:*

$$\begin{array}{ccc} Sig[P] \cup (Sig[SP] \cap Sig[SP_1]) & \xrightarrow{\subseteq} & Sig[SP] \\ {\scriptstyle v+\mathrm{id}}\downarrow & & \downarrow{\scriptstyle v_1} \\ Sig[SP_1] & \xrightarrow[i]{} & \Sigma'_1 \end{array}$$

2. *We define*

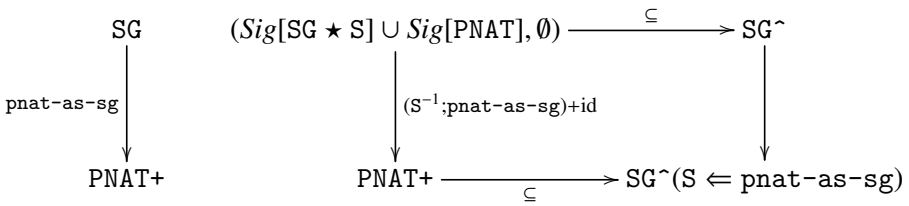
$$SP(P \Leftarrow v) = (SP \star v_1) \cup (SP_1 \star i).$$

In the actual situations when P is the renaming via an isomorphism p of another specification P_0 we specify a specification morphism $v_0 \colon P_0 \to SP_1$, usually called *view* in the literature. In this case of course the specification morphism v above is just $p^{-1}; v_0$ and the result $SP(P \Leftarrow v)$ of the instantiation may be denoted by $SP(p \Leftarrow v_0)$, a convention that is used by the OBJ family of languages. In most situations we may choose the result of the instantiation such that the underlying signature morphism of i is an inclusion; the explanation for this is given by Proposition 5.2 below.

Example 5.2. The multiplication of natural numbers may be specified as follows by using an instantiation of `SG^` by the signature morphism `pnat-as-sg`.

```
view pnat-as-sg from SG to PNAT+ {
  sort Elt -> PNat,
  op _+_ -> _+_
}
mod* PNAT* {
  protecting(SG^ (S <= pnat-as-sg) * {op _^_ -> _*_})
  eq M:PNat * 0 = 0 .
}
```

Then SG^(S⇐ pnat-as-sig) is obtained by the pushout of specification shown below:



Now we may define a specification morphism from SG to PNAT* that maps _+_ to _*_, called pnat*-as-sg. This requires the proof of the associativity of multiplication of natural numbers as inductive property. We skip this here. The following defines the power operation on the natural numbers.

```
mod* PNAT^ {
  protecting(SG^ (S <= pnat*-as-sg))
  eq M:PNat ^ 0 = s 0 .
}
```

Proposition 2.3 helps with providing the following alternative definition (Proposition 5.2 below) of instantiation of parameters that compared to Definition 5.2 has the disadvantage of being less intuitive but has the advantage of being technically more convenient in some situations. The additional technical condition given by Definition 2.10 helps with narrowing the class of possible isomorphic results of the instantiation. This condition holds quite naturally for certain classes of signature morphisms in many actual institutions through the pattern shown in the proof of the result below.

Proposition 5.1 (Free Extensions of **MSA** Signature Endo-Morphisms)**.** *In* **MSA** *every signature morphism* $\varphi\colon (S,F) \to (S,F)$ *strongly admits free extensions* φ' *along any inclusion of signatures* $(S,F) \subseteq (S',F')$.

Proof. Let us consider a fixed **MSA** signature $\Sigma_0 = (S_0, F_0)$.

At the level of the sort symbols, the free extension $\varphi'^{\mathrm{st}}\colon S' \to S'$ of φ^{st} along $S \subseteq S'$ is given by Fact 2.2:

$$\varphi'^{\mathrm{st}}(s') = \begin{cases} \varphi^{\mathrm{st}}(s') & \text{when } s' \in S, \\ s' & \text{otherwise.} \end{cases}$$

For each $(w_1, s_1) \in S'^* \times S'$, let us define the following three disjoint unions of sets:

$$\biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s} = \begin{cases} \emptyset & \text{when } (w_1,s_1) \notin S^* \times S, \\ \{(\sigma, ws, \Sigma_0) \mid \sigma \in F_{w\to s},\ \varphi^{\mathrm{st}}(ws) = w_1s_1\} & \\ & \text{when } (w_1,s_1) \in S^* \times S \text{ and } \varphi^{\mathrm{st}}(w_1s_1) \neq w_1s_1, \\ \{(\sigma, ws, \Sigma_0) \mid \sigma \in F_{w\to s},\ \varphi^{\mathrm{st}}(ws) = w_1s_1,\ ws \neq w_1s_1\} \cup F_{w_1\to s_1} & \\ & \text{when } (w_1,s_1) \in S^* \times S \text{ and } \varphi^{\mathrm{st}}(w_1s_1) = w_1s_1 \end{cases}$$

$$\biguplus_{ws=w_1s_1} F_{w\to s} = \begin{cases} \emptyset & \text{when } (w_1,s_1) \notin S^* \times S, \\ F_{w_1\to s_1} & \text{when } (w_1,s_1) \in S^* \times S \end{cases}$$

$$\biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'} = \begin{cases} \{(\sigma, w's', \Sigma_0) \mid \sigma \in F'_{w'\to s'},\ \varphi'^{\mathrm{st}}(w's') = w_1s_1\} & \\ & \text{when } \varphi'^{\mathrm{st}}(w_1s_1) \neq w_1s_1, \\ \{(\sigma, w's', \Sigma_0) \mid \sigma \in F'_{w'\to s'},\ \varphi'^{\mathrm{st}}(w's') = w_1s_1,\ w's' \neq w_1s_1\} \cup F'_{w_1\to s_1} & \\ & \text{when } \varphi'^{\mathrm{st}}(w_1s_1) = w_1s_1. \end{cases}$$

Note that $\biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s} \subseteq \biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'}$.
We define the function $\theta_{w_1\to s_1} : \biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s} \to \biguplus_{ws=w_1s_1} F_{w\to s}$ by

- $\theta_{w_1\to s_1} = \emptyset$, i.e. the empty function, when $(w_1, s_1) \notin S^* \times S$, and
- $\theta_{w_1\to s_1}(\sigma[, ws, \Sigma_0]) = \varphi^{\mathrm{op}}_{w\to s}(\sigma)$ when $(w_1, s_1) \in S^* \times S$.

Let us check that the condition of Fact 2.2 holds, namely that $\biguplus_{ws=w_1s_1} F_{w\to s}$ and $\biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'} \setminus \biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s}$ are disjoint, or equivalently that $\biguplus_{ws=w_1s_1} F_{w\to s} \cap \biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'}$ is a subset of $\biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s}$. If $\sigma \in \biguplus_{ws=w_1s_1} F_{w\to s} \cap \biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'}$ then $(w_1, s_1) \in S^* \times S$ and $\varphi'^{st}(w_1s_1) = w_1s_1$. We conclude that $\varphi^{st}(w_1s_1) = w_1s_1$, and therefore $\sigma \in \biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s}$.
Hence we consider the following free extension in $\mathbb{S}et$ given by Fact 2.2:

$$\begin{array}{ccc} \biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s} & \xrightarrow{\subseteq} & \biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'} \\ \Big\downarrow{\scriptstyle \theta_{w_1\to s_1}} & & \Big\downarrow{\scriptstyle \theta'_{w_1\to s_1}} \\ \biguplus_{ws=w_1s_1} F_{w\to s} & \xrightarrow[\subseteq]{} & (F'_1)_{w_1\to s_1} \end{array}$$

Note that according to Fact 2.2

$$(F'_1)_{w_1\to s_1} = \left(\biguplus_{ws=w_1s_1} F_{w\to s} \right) \cup \left(\biguplus_{\varphi'^{\mathrm{st}}(w's')=w_1s_1} F'_{w'\to s'} \setminus \biguplus_{\varphi^{\mathrm{st}}(ws)=w_1s_1} F_{w\to s} \right). \tag{30}$$

For any $w'' \in S'^*$ and $s'' \in S'$ we define $\varphi'_{w''\to s''}$ as the composition between the canonical injection $F'_{w''\to s''} \to \biguplus_{\varphi'^{\mathrm{st}}(w's')=\varphi'^{\mathrm{st}}(w''s'')} F'_{w'\to s'}$ and $\theta'_{\varphi'^{\mathrm{st}}(w'')\to\varphi'^{\mathrm{st}}(s'')}$.

We have thus obtained the inclusion of the signatures $(S, F) \subseteq (S', F'_1)$ and the morphism $\varphi' : (S', F') \to (S', F'_1)$. Our construction has followed the general construction of pushouts of **MSA** signature morphisms from pushouts of functions (see [30, 12]); therefore, the square depicted below is a pushout square.

$$\begin{array}{ccc} (S, F) & \xrightarrow{\subseteq} & (S', F') \\ \Big\downarrow{\scriptstyle \varphi} & & \Big\downarrow{\scriptstyle \varphi'} \\ (S, F) & \xrightarrow[\subseteq]{} & (S', F'_1) \end{array}$$

In order to show that φ' is a free extension of φ let us consider a signature $(\overline{S}, \overline{F})$ such that for any sort or operation symbol x in $(\overline{S}, \overline{F}) \cap (S, F)$ we have that $\varphi(x) = x$. Since φ'^{st} is a free extension of φ^{st} we immediately obtain that $\varphi'^{\mathrm{st}}(s) = s$, for every $s \in \overline{S} \cap S$. Let us now consider $\sigma \in F'_{w\to s} \cap (\overline{F})_{w\to s}$ for some $(w, s) \in (S' \cap \overline{S})^* \times (S' \cap \overline{S})$. We have two situations:

1. $\sigma \in F_{w\to s}$: in this case we have that $\varphi'^{\mathrm{op}}_{w\to s}(\sigma) = \varphi^{\mathrm{op}}_{w\to s}(\sigma)$; since σ belongs $(S, F) \cap (\overline{S}, \overline{F})$ we also have that $\varphi^{\mathrm{op}}_{w\to s}(\sigma) = \sigma$ hence $\varphi'^{\mathrm{op}}_{w\to s}(\sigma) = \sigma$.
2. $\sigma \notin F_{w\to s}$: in this case since $(w, s) \in (S' \cap \overline{S})^* \times (S' \cap \overline{S})$ we have that $\varphi'^{\mathrm{st}}(ws) = ws$; hence the canonical injection $F'_{w\to s} \to \biguplus_{\varphi'^{\mathrm{st}}(w's')=ws} F'_{w'\to s'}$ is inclusion and also by Fact 2.2 we have that $\theta'_{w\to s}(\sigma) = \sigma$. Consequently $\varphi'^{\mathrm{op}}_{w\to s}(\sigma) = \sigma$.

In order to complete our argument it remains to show that

$$(S_0, F_0) \cap (S', F'_1) \subseteq (S_0, F_0) \cap (S', F').$$

At the level of the sort symbols the above relation is trivial. Let $x \in (F_0)_{w_1\to s_1} \cap (F'_1)_{w_1\to s_1}$ for some fixed $(w_1, s_1) \in (S' \cap S_0)^* \times (S' \cap S_0)$. Let us recall the value of $(F'_1)_{w_1\to s_1}$ given by (30). By set theoretic arguments, since x is in Σ_0, it cannot be of the form (σ, ws, Σ_0). It follows that $x \in F'_{w_1\to s_1}$. □

Proposition 5.2. *The instantiation of a parameterized specification as defined by Definition 5.2 may be obtained by a pushout of specification morphisms as follows:*

$$\begin{array}{ccc} P \cup SP_1 & \xrightarrow{\subseteq} & SP \cup SP_1 \\ {\scriptstyle v+1_{Sig[SP_1]}}\downarrow & & \downarrow{\scriptstyle v'} \\ SP_1 & \xrightarrow[i]{} & SP(P \Leftarrow v) \end{array}$$

Moreover if in addition

1. *the inclusion system for the signatures is epic, and*
2. *each idempotent-by-extension signature morphism admits free extensions along any signature inclusion (with the same domain)*

then we may choose $SP(P \Leftarrow v)$ such that $Sig[SP_1] \subseteq Sig[SP(P \Leftarrow v)]$.

Proof. For the first part of the proposition we apply Proposition 2.3 for the diagram below of signature morphisms.

$$\begin{array}{ccccc} Sig[P] \cup (Sig[SP] \cap Sig[SP_1]) & \xrightarrow[j]{\subseteq} & Sig[P \cup SP_1] & \xrightarrow{v+1} & Sig[SP_1] \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} & & \downarrow{\scriptstyle i} \\ Sig[SP] & \xrightarrow[\subseteq]{j_1} & Sig[SP \cup SP_1] & \xrightarrow[v']{} & Sig[SP(P \Leftarrow v)] \end{array}$$

The conclusion for this part follows now by the calculation below that uses some of the equations of Proposition 4.1, Fact 4.1 and Proposition 4.2.

$$\begin{aligned} &(SP_1 \star i) \cup ((SP \cup SP_1) \star v_1) \\ &\quad \amalg (SP_1 \star i) \cup SP \star ((Sig[SP] \subseteq Sig[SP \cup SP_1]); v') \cup SP_1 \star ((Sig[SP_1] \subseteq Sig[SP \cup SP_1]); v') \\ &\quad \amalg (SP_1 \star i) \cup (SP \star (j_1; v')) \cup SP_1 \star ((Sig[SP_1] \subseteq Sig[P \cup SP_1]); (Sig[P \cup SP_1] \subseteq Sig[SP \cup SP_1]); v') \\ &\quad \amalg (SP_1 \star i) \cup (SP \star (j_1; v')) \cup (SP_1 \star i) \\ &\quad \amalg (SP_1 \star i) \cup (SP \star (j_1; v')). \end{aligned}$$

For the second part of the proposition we apply the first part of Proposition 2.4 for $Sig[P] \cup Sig[SP_1]$ in the role of A and $Sig[SP] \cup Sig[SP_1]$ in the role of B.

$$\begin{array}{ccccc} Sig[P \cup SP_1] & \xrightarrow{\nu+1} & Sig[SP_1] & \xrightarrow{\subseteq} & Sig[P \cup SP_1] \\ \downarrow{\scriptstyle \subseteq} & & \downarrow{\scriptstyle \subseteq\, i} & & \downarrow{\scriptstyle \subseteq} \\ Sig[SP \cup SP_1] & \xrightarrow[\nu']{} & Sig[SP(P \Leftarrow \nu)] & \xrightarrow[\subseteq]{} & \Sigma_1' \end{array}$$

The role of the arrow f of Proposition 2.4 is played by the composition $(\nu + 1_{Sig[SP_1]}); (Sig[SP_1] \subseteq Sig[P_1] \cup Sig[SP_1])$; it is easy to see that f is idempotent-by-extension and that $e_f = \nu + 1_{Sig[SP_1]}$ (because this is a retract and from [14] we know that each retract is an abstract surjection). By the assumption on the existence of free extensions we get the g of Proposition 2.4 and by the first part of Proposition 2.4 we may define ν' as e_g. □

When the inclusion system for the signatures is epic and each idempotent-by-extension signature morphism admits free extension we shall always implicitly assume that $SP(P \Leftarrow \nu)$ is chosen such that

$$Sig[SP_1] \subseteq Sig[SP(P \Leftarrow \nu)].$$

Example 5.3. MSA fulfils the additional conditions of Proposition 5.2, the conditions on the existence of free extensions being a special case of Proposition 5.1. Hence for the structured specifications over **MSA** we may use the alternative definition of parameter instantiation given by Proposition 5.2. For example, the instantiation `SG^(S⇐ pnat-as-sig)` of Example 5.2 may be obtained by the following pushout of specifications:

$$\begin{array}{ccc} \texttt{(SG} \star \texttt{S)} \cup \texttt{(PNAT+)} & \xrightarrow{\subseteq} & \texttt{(SG\^{})} \cup \texttt{(PNAT+)} \\ \downarrow{\scriptstyle (\texttt{S}^{-1};\texttt{pnat-as-sg})+1_{\texttt{PNAT+}}} & & \downarrow \\ \texttt{PNAT+} & \xrightarrow[\subseteq]{} & \texttt{SG\^{}(S} \Leftarrow \texttt{pnat-as-sg)} \end{array}$$

5.2. Multiple parameters

Specification modules may sometimes contain more than one parameter as in the example below.

Example 5.4. This is an example of a parameterized specification of the mathematical concept of semigroup homomorphism that uses two semigroup parameters, one for the source, and the other for the target of the homomorphism.

```
mod* SGH (S1 :: SG, S2 :: SG) {
  op h : Elt.S1 -> Elt.S2
  vars X Y : Elt.S1
  eq h(X + Y) = h(X) + h(Y) .
}
```

Here we use two parameters based upon the same specification, namely `SG`. In general, this need not be the case.

The general condition of multiple parameters is that any two parameters of the same parameterized specification should be disjoint.

Definition 5.3 (Multiple Parameters)**.** A *multiple parameterized specification* is a specification with several parameters such that for any parameters P_1 and P_2 we have that $Sig[P_1]$ and $Sig[P_2]$ are disjoint.

Example 5.5. This condition is easily guaranteed in the language CafeOBJ by the qualification system corresponding to the parameters. This can be noticed in `SGH` where the sorts of `S1`, resp. `S2`, are denoted by `Elt.S1`, resp. `Elt.S2`.

Proposition 5.3. *For any multiple parameterized specification $SP(P_1, \ldots, P_n)$ we have that $SP(P_1 \cup \cdots \cup P_n)$ is a (single) parameterized specification.*

Proof. It is enough to do this for $n = 2$, since this can be immediately extended to greater n by induction.

Because each $Sig[P_i] \subseteq Sig[P_1 \cup P_2]$ we have that $Sig[P_1 \cup P_2] = Sig[P_1] \cup Sig[P_2] \subseteq Sig[SP]$. Moreover for each $M \in Mod[SP]$ we have that $M{\restriction}_{Sig[P_1 \cup P_2]}{\restriction}_{Sig[P_i]} = M{\restriction}_{Sig[P_i]} \in Mod[P_i]$. Hence $M{\restriction}_{Sig[P_1 \cup P_2]} \in Mod[P_1 \cup P_2]$ which shows that the signature inclusion $Sig[P_1 \cup P_2] \subseteq Sig[SP]$ is a specification morphism $P_1 \cup P_2 \to SP$. □

The definition of the simultaneous instantiation of multiple parameters is just a special case of the definition Definition 5.2 of the instantiation of a single parameter as follows. For the sake of simplicity of the presentation we consider the simplest case, that of two parameters, the general case getting the same treatment.

Corollary 5.2 (Simultaneous Instantiation of Parameters)**.** *Let us consider a multiple parameterized specification $SP(P_1, P_2)$ with two parameters P_1 and P_2. Then for any specification morphisms $v_1 : P_1 \to SP_1$ and $v_2 : P_2 \to SP_2$ such that for all $i, j \in \{1, 2\}$ $Sig[P_i]$ and $Sig[SP_j]$ are disjoint, we have that $P_1 \cup P_2$ and $SP_1 \cup SP_2$ are disjoint.*

Consequently, since the condition of Definition 5.2 is fulfilled, the instance of $SP(P_1, P_2)$ by v_1 and v_2 *is defined as $SP(P_1 \cup P_2 \Leftarrow v_1 + v_2)$ where $v_1 + v_2$ is the unique specification morphism that makes the diagram below commute*

$$\begin{array}{ccccc} P_1 & \xrightarrow{\subseteq} & P_1 \cup P_2 & \xleftarrow{\supseteq} & P_2 \\ {\scriptstyle v_1}\downarrow & & \downarrow{\scriptstyle v_1+v_2} & & \downarrow{\scriptstyle v_2} \\ SP_1 & \xrightarrow[\subseteq]{} & SP_1 \cup SP_2 & \xleftarrow[\supseteq]{} & SP_2 \end{array}$$

Proof. That $Sig[P_1 \cup P_2]$ and $Sig[SP_1 \cup SP_2]$ are disjoint follows from Proposition 2.2 (applied twice).

Because $Sig[P_1]$ and $Sig[P_2]$ are disjoint, by Corollary 2.1 we have that $Sig[P_1 \cup P_2]$ is their coproduct. Therefore there exists an unique signature morphism $(v_1 + v_2) : Sig[P_1 \cup P_2] \to Sig[SP_1 \cup SP_2]$ such that $v_i ; (Sig[SP_i] \subseteq Sig[SP_1 \cup SP_2]) = (Sig[P_i] \subseteq Sig[P_1 \cup P_2]) ; (v_1 + v_2)$. Moreover by (10) and (6) we have that

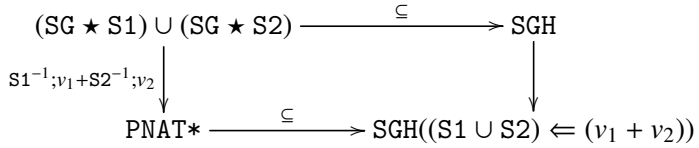
$$(P_1 \cup P_2) \star (v_1 + v_2) = P_1 \star v_1 \cup P_2 \star v_2$$

and since $SP_i \models P_i \star v_i$ (because v_i are specification morphisms) we obtain $SP_1 \cup SP_2 \models (P_1 \cup P_2) \star (v_1 + v_2)$ which shows that $v_1 + v_2$ is indeed a specification morphism $P_1 \cup P_2 \to SP_1 \cup SP_2$. □

Example 5.6. We can obtain the powers of any natural number by instantiating the semigroup homomorphism specification as follows:

```
mod* PNATn {
  protecting(PNAT)
  op n : -> PNat
}
mod! POWERofN {
  protecting(PNATn)
  protecting(SG-HOM(S1 <= view to PNAT+ { op _+_ -> _+_},
  S2 <= view to PNAT* { op _+_ -> _*_}))
  eq h(0) = s 0 .
  eq h(s 0) = n .
}
```

The result of the instantiation imported by `POWERofN` is explained by Corollary 5.2, with the corresponding pushout diagram being as follows:



5.3. Sequential instantiation of parameters

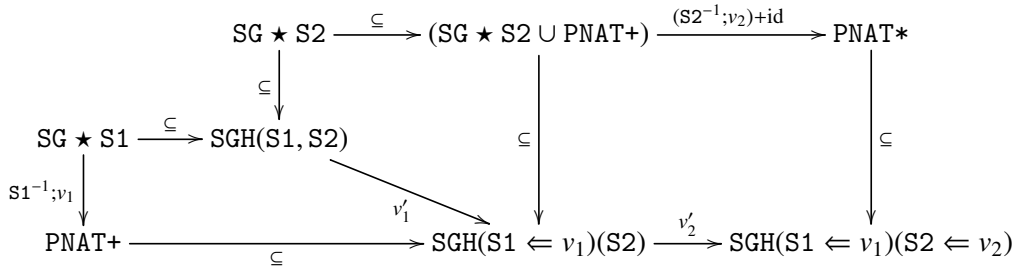
Example 5.7. The result of `POWERofN` may be obtained in a different way, namely by instantiating the parameters `S1` and `S2` one by one as follows.

```
mod! POWERofN {
  protecting(PNATn)
  protecting(SG-HOM(S1 <= view to PNAT+ { op _+_ -> _+_})
  (S2 <= view to PNAT* { op _+_ -> _*_}))
  eq h(0) = s 0 .
  eq h(s 0) = n .
}
```

This means that

1. we instantiate the first parameter `S1` and obtain a parameterized module $\texttt{SGH}(\texttt{S1} \Leftarrow v_1)(\texttt{S2})$ (where v_1 is the view corresponding to `S1`), and
2. we instantiate `S2` and obtain the final result $\texttt{SGH}(\texttt{S1} \Leftarrow v_1)(\texttt{S2} \Leftarrow v_2)$ (where v_2 is the view corresponding to `S2`).

This process can be seen in the diagram below.



Note that as the result of the first instantiation step PNAT+ has to be shared with the instance of the second parameter, hence according to Definition 5.2 the specification SG ⋆ S2 ∪ PNAT+ appears in the pushout of the second instantiation.

Another point is that $Sig[\mathtt{SG} \star \mathtt{S2}]$ is included in $Sig[\mathtt{SGH}(\mathtt{S1} \Leftarrow v_1)(\mathtt{S2})]$ hence (S2::SG) can be regarded as a parameter for $\mathtt{SGH}(\mathtt{S1} \Leftarrow v_1)(\mathtt{S2})$ in the sense of Definition 5.2.

The following is the general procedure of sequential instantiation of parameters. Given the data of Corollary 5.2 we instantiate the parameters one by one by treating them as single separate parameters (Definition 5.2). Because in this case it is technically more convenient, let us use the variant of parameter instantiation given by Proposition 5.2. The process of sequential instantiation of parameters can be visualized in the diagram below:

$$\begin{array}{ccccc}
 & & P_1 \cup SP_1 & \xrightarrow{v_1 + 1_{Sig[SP_1]}} & SP_1 \\
 & & \subseteq \downarrow i_1 & & i_1' \downarrow \subseteq \\
P_2 & \xrightarrow{\subseteq} & SP \cup SP_1 & \xrightarrow{v_1'} & SP(P_1 \Leftarrow v_1) \\
\subseteq \downarrow i_2 & & & & i_2' \downarrow \subseteq \\
P_2 \cup SP_2 & & \xrightarrow[i_3]{\subseteq} & & SP(P_1 \Leftarrow v_1) \cup SP_2 \\
v_2 + 1_{Sig[SP_2]} \downarrow & & & & \downarrow v_2' \\
SP_2 & & \xrightarrow[\subseteq]{i_3'} & & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)
\end{array} \tag{31}$$

The correctness of the second instantiation step relies upon the fact that P_2 is indeed a parameter for the result $SP(P_1 \Leftarrow v_1)$ of the first instantiation step. This follows immediately from the result below.

Proposition 5.4. *In addition to the technical hypotheses underlying the sequential instantiation defined above let us also assume that*

- *there exists a signature* 0 *initial both in* $\mathbb{S}ig$ *and in the subcategory* $\mathcal{I}$ *of the abstract inclusions,*
- *the inclusion system is epic and distributive, and*
- *each idempotent-by-extension signature morphism admits free extensions.*

Then for the diagram of sequential instantiation (31) the signature morphism $(Sig[P_2] \subseteq Sig[SP \cup SP_1]); v_1'$ *is an inclusion.*

Proof. We apply Proposition 2.4 for $Sig[P_1 \cup SP_1]$ in the role of A, $Sig[SP \cup SP_1]$ in the role of B, and $Sig[P_2]$ in the role of the preserved object. By Proposition 2.2 we have that $Sig[P_2]$ and $Sig[P_1 \cup SP_1]$ are disjoint and by the assumption that there exists a signature 0 initial both in $\mathbb{S}ig$ and in the subcategory of the abstract inclusions, it is easy to see (by antisymmetry) that $Sig[P_2] \cap (Sig[P_1 \cup SP_1]) = 0$, hence $Sig[P_2] \cap (Sig[P_1 \cup SP_1]) \subseteq Sig[SP_1]$. It follows that

$$\begin{aligned}
&(Sig[P_2] \cap (Sig[P_1 \cup SP_1]) \subseteq Sig[SP \cup SP_1]); (v_1 + 1_{Sig[SP_1]}) \\
&\quad = (Sig[P_2] \cap (Sig[P_1 \cup SP_1]) \subseteq Sig[SP_1]); (Sig[SP_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}) \\
&\quad = (Sig[P_2] \cap (Sig[P_1 \cup SP_1]) \subseteq Sig[SP_1]); 1_{Sig[SP_1]} \\
&\quad = (Sig[P_2] \cap (Sig[P_1 \cup SP_1]) \subseteq Sig[SP_1]).
\end{aligned}$$

This together with the condition on the existence of free extensions allows us to apply the second part of Proposition 2.4 in order to get that $(Sig[P_2] \subseteq Sig[SP \cup SP_1]); v_1'$ is an inclusion. □

Note that the additional condition on the existence of the signature 0 holds naturally in the examples, in the case of **MSA** the signature 0 being just the empty signature.

Theorem 5.1. *Let $SP(P_1 \cup P_2 \Leftarrow v_1 + v_2)$ and $SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$ be two instances of a multiple parameterized specification $SP(P_1, P_2)$. Under the conditions of Proposition 5.4 the simultaneous and the sequential instantiation of multiple parameters are isomorphic, provided that $SP(P_1 \Leftarrow v_1)$ can be chosen such that $Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP \cup SP_1]$. More precisely, there exists an isomorphism of specifications such that the diagram below commutes*

$$\begin{array}{ccc} SP(P_1 \cup P_2 \Leftarrow v_1 + v_2) & \xrightarrow{\cong} & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \\ & {\scriptstyle\subseteq}\nwarrow \quad \nearrow{\scriptstyle\subseteq} & \\ & SP_1 \cup SP_2 & \end{array}$$

Proof. Consider the instantiation diagram (31) above. Let us first show that $v'_1; i'_2; v'_2$ and i'_3 are compatible. By the definition of compatibility this means showing that

$$(Sig[SP \cup SP_1] \cap Sig[SP_2] \subseteq Sig[SP \cup SP_1]); v'_1; i'_2; v'_2 \text{ is an inclusion.} \tag{32}$$

This is done in two steps. First we show that

$$(Sig[SP \cup SP_1] \cap Sig[SP_2] \subseteq Sig[SP \cup SP_1]); v'_1 \text{ is an inclusion.} \tag{33}$$

For this we apply (the second part of) Proposition 2.4 for the pushout square defining $SP(P_1 \Leftarrow v_1)$. A straightforward calculation shows that $(Sig[SP \cup SP_1] \cap Sig[SP_2]) \cap Sig[P_1 \cup SP_1] = Sig[SP_1] \cap Sig[SP_2]$. We also have that

$$\begin{aligned} &(Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}) \\ &\quad = (Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_1]); (Sig[SP_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}) \\ &\quad = (Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_1]); 1_{Sig[SP_1]} = Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_1] \end{aligned}$$

which allows us to apply Proposition 2.4 for obtaining (33).
Now the conclusion (32) is obtained by applying Proposition 2.4 to the pushout square defining the instantiation $SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$. Since $(Sig[SP \cup SP_1] \cap Sig[SP_2]) \cap Sig[P_2 \cup SP_2] = Sig[SP \cup SP_1] \cap Sig[SP_2]$ we can apply Proposition 2.4 as follows

$$\begin{aligned} &(Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}) \\ &\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2]); (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}) \\ &\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2]); 1_{Sig[SP_2]} \\ &\quad = Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2]. \end{aligned}$$

The next step in our proof is to establish that

$$i'_1; i'_2; v'_2 = (Sig[SP_1] \subseteq Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)]). \tag{34}$$

This is achieved through the application of the second part of Proposition 2.4 to the pushout square defining $SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$ justified by noting that

$$Sig[SP_1] \cap Sig[P_2 \cup SP_2] = Sig[SP_1] \cap Sig[SP_2]$$

and by the following calculation

$$
\begin{aligned}
&(Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}) \\
&\quad = (Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_2]); (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}) \\
&\quad = (Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_2]); 1_{Sig[SP_2]} \\
&\quad = Sig[SP_1] \cap Sig[SP_2] \subseteq Sig[SP_2].
\end{aligned}
$$

The relations (32) (giving the compatibility between $v'_1; i'_2; v'_2$ and i'_3) and (34) together with the fact that $Sig[SP_2] \subseteq Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)]$ (giving the compatibility between $i'_1; i'_2; v'_2$ and i'_3) allow us to draw the following square of specification morphisms:

$$
\begin{array}{ccc}
P_1 \cup P_2 \cup SP_1 \cup SP_2 & \xrightarrow[j]{\subseteq} & SP \cup SP_1 \cup SP_2 \\
\Big\downarrow{\scriptstyle v_1+v_2+1_{Sig[SP_1 \cup SP_2]}} & & \Big\downarrow{\scriptstyle (v'_1;i'_2;v'_2) \vee i'_3} \\
SP_1 \cup SP_2 & \xrightarrow[\subseteq]{i=(i'_1;i'_2;v'_2) \vee i'_3} & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)
\end{array}
$$

The conclusion of our theorem follows once we have proved that this is a pushout square. Let us first show that it is commutative. For this we use the pushout property of unions for $Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]$ which means that it is enough to check the restriction of the commutativity property of the diagram to each of the four components of this union as follows:

$$
\begin{aligned}
&(Sig[P_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) \\
&\quad = ((Sig[P_1] \subseteq Sig[P_1 \cup SP_1]); i_1); v'_1; i'_2; v'_2 \\
&\quad = (Sig[P_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}); i'_1; i'_2; v'_2 \\
&\quad = v_1; i'_1; i'_2; v'_2 = v_1; (Sig[SP_1] \subseteq Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)]) \text{ (by (34))} \\
&\quad = v_1; (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); i = (Sig[P_1] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); i.
\end{aligned}
$$

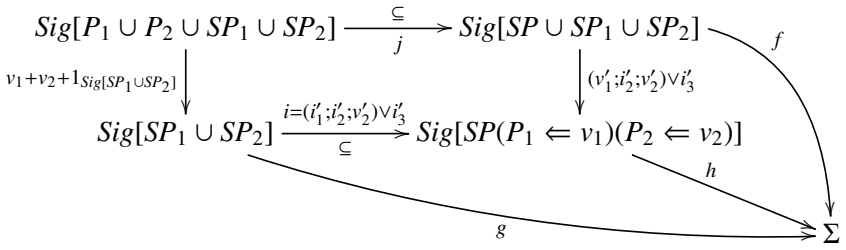
$$
\begin{aligned}
&(Sig[P_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) \\
&\quad = (Sig[P_2] \subseteq Sig[SP \cup SP_1]); (Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) \\
&\quad = (Sig[P_2] \subseteq Sig[SP \cup SP_1]); (v'_1; i'_2; v'_2) = i_2; i_3; v'_2 = i_2; (v_2 + 1_{Sig[SP_2]}); i'_3 = v_2; i'_3 \\
&\quad = (Sig[P_2] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); i.
\end{aligned}
$$

$$
\begin{aligned}
&(Sig[SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) \\
&\quad = (Sig[SP_1] \subseteq Sig[SP \cup SP_1]); (Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) \\
&\quad = (Sig[SP_1] \subseteq Sig[SP \cup SP_1]); (v'_1; i'_2; v'_2) \\
&\quad = (Sig[SP_1] \subseteq Sig[P_1 \cup SP_1]); ((Sig[P_1 \cup SP_1] \subseteq Sig[SP \cup SP_1]); v'_1); i'_2; v'_2 \\
&\quad = (Sig[SP_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}); i'_1; i'_2; v'_2 = 1_{Sig[SP_1]}; i'_1; i'_2; v'_2 = i'_1; i'_2; v'_2 \\
&\quad = (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); i = (Sig[SP_1] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); i.
\end{aligned}
$$

$$
\begin{aligned}
&(Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3) = i'_3 = (Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); i \\
&\quad = (Sig[SP_2] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); i.
\end{aligned}
$$

Now we show that this commutative square is a pushout. For this we apply Proposition 3.3. Thus at the first stage we show that the underlying square of signature morphisms is a pushout. Let us consider signature morphisms $f\colon Sig[SP \cup SP_1 \cup SP_2] \to \Sigma$ and $g\colon Sig[SP_1 \cup SP_2] \to \Sigma$ such that $j; f = (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); g$. We have to show that there exists an unique signature morphism $h\colon Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)] \to \Sigma$ such that

$$((v'_1; i'_2; v'_2) \vee i'_3); h = f \quad \text{and} \quad i; h = g. \tag{35}$$



Let us introduce the following notations:

- $f_1 = (Sig[SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f$,
- $f_2 = (Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f$,
- $f_0 = (Sig[SP] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f$.
- $g_1 = (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); g$, and
- $g_2 = (Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); g$.

Note that $j; f = (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); g$ implies that

$$f_1 = g_1 \quad \text{and} \quad f_2 = g_2. \tag{36}$$

Since f_0 and f_1 are compatible let us we consider $f_0 \vee f_1\colon Sig[SP \cup SP_1] \to \Sigma$. Let us show that

$$i_1; (f_0 \vee f_1) = (v_1 + 1_{Sig[SP_1]}); f_1. \tag{37}$$

By the pushout property of the union $Sig[P_1 \cup SP_1]$ it is enough to perform the following two calculations:

$$\begin{aligned}
&(Sig[P_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}); f_1 = v_1; f_1 = v_1; g_1 = v_1; (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); g \\
&\quad = (Sig[P_1] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); g \\
&\quad = (Sig[P_1] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); j; f \\
&\quad = (Sig[P_1] \subseteq Sig[SP \cup SP_1]); (f_0 \vee f_1) = (Sig[P_1] \subseteq Sig[P_1 \cup SP_1]); i_1; (f_0 \vee f_1).
\end{aligned}$$

$$\begin{aligned}
&(Sig[SP_1] \subseteq Sig[P_1 \cup SP_1]); (v_1 + 1_{Sig[SP_1]}); f_1 = 1_{Sig[SP_1]}; f_1 = f_1 \\
&\quad = (Sig[SP_1] \subseteq Sig[SP_1 \cup SP]); (f_0 \vee f_1) = (Sig[P_1] \subseteq Sig[P_1 \cup SP_1]); i_1; (f_0 \vee f_1).
\end{aligned}$$

From (37) and the pushout property of the square defining $SP(P_1 \Leftarrow v_1)$ there exists an unique morphism $q\colon Sig[SP(P_1 \Leftarrow v_1)] \to \Sigma$ such that $v'_1; q = f_0 \vee f_1$ and $i'_1; q = f_1$.

Now let us show that q and f_2 are compatible. On the one hand we know from the hypotheses that $Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP \cup SP_1]$, hence $Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP_2] \cap Sig[SP \cup$

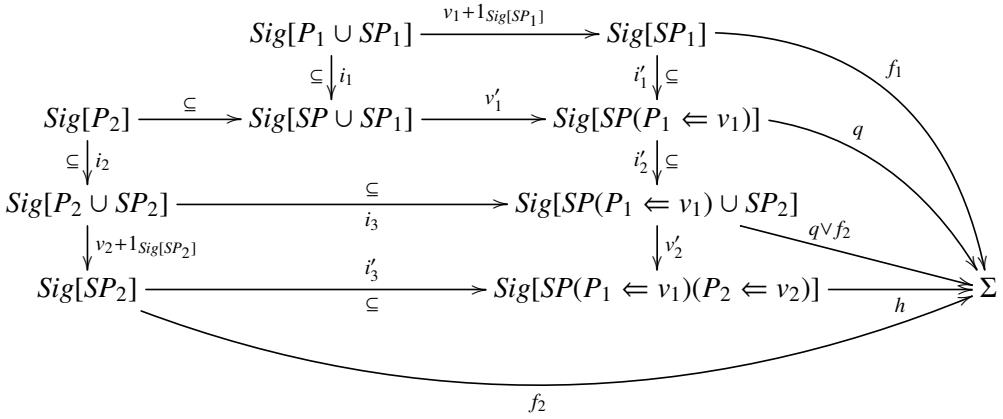
$SP_1]$. On the other hand from (33) we have that $Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP(P_1 \Leftarrow v_1)]$ hence $Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)]$. Thus

$$Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] = Sig[SP_2] \cap Sig[SP \cup SP_1].$$

Based on this relation we have the following calculation showing the compatibility between f_2 and q.

$$\begin{aligned}
&(Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP(P_1 \Leftarrow v_1)]); q \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP(P_1 \Leftarrow v_1)]); q \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1]); v_1'; q \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1]); (f_0 \vee f_1) \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2]); (Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f \\
&\quad = (Sig[SP_2] \cap Sig[SP \cup SP_1] \subseteq Sig[SP_2]); f_2 = (Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP_2]); f_2.
\end{aligned}$$

Therefore let $q \vee f_2 : Sig[SP(P_1 \Leftarrow v_1) \cup SP_2] \to \Sigma$.



Let us show that

$$i_3; (q \vee f_2) = (v_2 + 1_{Sig[SP_2]}); f_2. \tag{38}$$

By the pushout property of the union $Sig[P_2 \cup SP_2]$ for showing (38) it is enough to perform the following calculations:

$$\begin{aligned}
&(Sig[P_2] \subseteq Sig[P_2 \cup SP_2]); i_3; (q \vee f_2) = (Sig[P_2] \subseteq Sig[SP \cup SP_1]); v_1'; i_2'; (q \vee f_2) \\
&\quad = (Sig[P_2] \subseteq Sig[SP \cup SP_1]); v_1'; q = (Sig[P_2] \subseteq Sig[SP \cup SP_1]); (f_0 \vee f_1) \\
&\quad = (Sig[P_2] \subseteq Sig[SP]); (Sig[SP] \subseteq Sig[SP \cup SP_1]); (f_0 \vee f_1) = (Sig[P_2] \subseteq Sig[SP]); f_0 \\
&\quad = (Sig[P_2] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); j; f \\
&\quad = (Sig[P_2] \subseteq Sig[P_1 \cup P_2 \cup SP_1 \cup SP_2]); (v_1 + v_2 + 1_{Sig[SP_1 \cup SP_2]}); g \\
&\quad = v_2; (Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); g = v_2; g_2 = v_2; f_2 \\
&\quad = (Sig[P_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}); f_2.
\end{aligned}$$

$$\begin{aligned}
&(Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); i_3; (q \vee f_2) = (Sig[SP_2] \subseteq Sig[SP(P_1 \Leftarrow v_1) \cup SP_2]); (q \vee f_2) \\
&\quad = f_2 = 1_{Sig[SP_2]}; f_2 = (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}); f_2.
\end{aligned}$$

Now from (38) and the pushout property of the square defining $SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$ there exists an unique $h\colon Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)] \to \Sigma$ such that $v'_2; h = q \vee f_2$ and $i'_3; h = f_2$. Let us prove that h satisfies the relations (35). For this we use the pushout properties of the unions $Sig[SP_1 \cup SP_2 \cup SP]$ and $Sig[SP_1 \cup SP_2]$, resp. Hence the proof of $((v'_1; i'_2; v'_2) \vee i'_3); h = f$ is achieved through the following couple of calculations:

$$\begin{aligned}&(Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3); h = v'_1; i'_2; v'_2; h = f_0 \vee f_1\\ &\quad = (Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f\end{aligned}$$

$$\begin{aligned}&(Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); ((v'_1; i'_2; v'_2) \vee i'_3); h = i'_3; h = f_2\\ &\quad = (Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]); f.\end{aligned}$$

and the proof of $i; h = g$ is achieved through the following couple of calculations:

$$\begin{aligned}&(Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); i; h = (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); ((i'_1; i'_2; v'_2) \vee i'_3); h\\ &\quad = i'_1; i'_2; v'_2; h = f_1 = g_1 = (Sig[SP_1] \subseteq Sig[SP_1 \cup SP_2]); g\end{aligned}$$

$$\begin{aligned}&(Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); i; h = (Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); ((i'_1; i'_2; v'_2) \vee i'_3); h\\ &\quad = i'_3; h = f_2 = g_2 = (Sig[SP_2] \subseteq Sig[SP_1 \cup SP_2]); g.\end{aligned}$$

For showing the uniqueness of h satisfying the relations (35) let us assume there exists another morphism $h'\colon Sig[SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)] \to \Sigma$ such that

$$((v'_1; i'_2; v'_2) \vee i'_3); h' = f \text{ and } ((i'_1; i'_2; v'_2) \vee i'_3); h' = g.$$

By left composition of the first equation with $Sig[SP \cup SP_1] \subseteq Sig[SP \cup SP_1 \cup SP_2]$ and with $Sig[SP_2] \subseteq Sig[SP \cup SP_1 \cup SP_2]$ we obtain

$$v'_1; i'_2; v'_2; h' = f_0 \vee f_1 \text{ and } i'_3; h' = f_2.$$

Since $v_1 + 1_{Sig[SP_1]}$ is epi (see the proof of Proposition 5.2) and epis are stable under pushouts, it follows that v'_1 is also epi. This means

$$i'_2; v'_2; h' = q \text{ and } i'_3; h' = f_2.$$

Since

$$\begin{aligned}&(Sig[SP_2] \subseteq Sig[SP(P_1 \Leftarrow v_1) \cup SP_2]); v'_2\\ &\quad = (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (Sig[P_2 \cup SP_2] \subseteq Sig[SP(P_1 \Leftarrow v_1) \cup SP_2]); v'_2\\ &\quad = (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]); (v_2 + 1_{Sig[SP_2]}); i'_3 = i'_3\end{aligned}$$

we obtain that

$$i'_2; v'_2; h' = q \text{ and } (Sig[SP_2] \subseteq Sig[SP(P_1 \Leftarrow v_1) \cup SP_2]); v'_2; h' = f_2.$$

By the uniqueness aspect of the pushout property of the union $Sig[SP(P_1 \Leftarrow v_1) \cup SP_2]$ we further obtain that $v'_2; h' = q \vee f_2$ which by the epi property of v'_2 implies $h' = h$.

In order to complete our proof that the square

$$\begin{array}{ccc} P_1 \cup P_2 \cup SP_1 \cup SP_2 & \xrightarrow[j]{\subseteq} & SP \cup SP_1 \cup SP_2 \\ \downarrow{\scriptstyle v_1+v_2+1_{Sig[SP_1 \cup SP_2]}} & & \downarrow{\scriptstyle (v_1';i_2';v_2')\vee i_3'} \\ SP_1 \cup SP_2 & \xrightarrow[\subseteq]{i=(i_1';i_2';v_2')\vee i_3'} & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \end{array}$$

is a pushout of specifications, according to the construction of pushouts of specifications given by Proposition 3.3 it remains to show that

$$SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \uplus (SP \cup SP_1 \cup SP_2) \star ((v_1'; i_2'; v_2') \vee i_3') \cup (SP_1 \cup SP_2) \star i. \tag{39}$$

The proof of (39) consists of the following calculation that uses the rules (3), (4), and (5) of Proposition 4.1, (6) of Fact 4.1, and (10) of Proposition 4.2:

$$\begin{aligned} & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \uplus SP_2 \star i_3' \cup (SP(P_1 \Leftarrow v_1) \cup SP_2) \star v_2' \\ & \quad \uplus SP_2 \star i_3' \cup SP(P_1 \Leftarrow v_1) \star i_2' \star v_2' \cup SP_2 \star (Sig[SP_2] \subseteq Sig[P_2 \cup SP_2]) \star i_3 \star v_2' \\ & \quad \uplus SP_2 \star i_3' \cup SP(P_1 \Leftarrow v_1) \star i_2' \star v_2' \cup SP_2 \star i_3' \\ & \quad \uplus SP_2 \star i_3' \cup SP(P_1 \Leftarrow v_1) \star i_2' \star v_2' \\ & \quad \uplus SP_2 \star i_3' \cup (SP_1 \star i_1' \cup SP \star Sig[SP \cup SP_1] \star v_1') \star i_2' \star v_2' \\ & \quad \uplus SP_2 \star i_3' \cup SP_1 \star (i_1'; i_2'; v_2') \cup SP \star Sig[SP \cup SP_1] \star (v_1'; i_2'; v_2') \end{aligned}$$

$$\begin{aligned} & (SP \cup SP_1 \cup SP_2) \star ((v_1'; i_2'; v_2') \vee i_3') \cup (SP_1 \cup SP_2) \star i \\ & \quad \uplus (SP \cup SP_1) \star (v_1'; i_2'; v_2') \cup SP_2 \star i_3' \cup SP_1 \star Sig[SP_1 \cup SP_2] \star i \cup SP_2 \star Sig[SP_1 \cup SP_2] \star i \\ & \quad \uplus SP \star Sig[SP \cup SP_1] \star (v_1'; i_2'; v_2') \cup \\ & \qquad SP_1 \star Sig[SP \cup SP_1] \star (v_1'; i_2'; v_2') \cup SP_2 \star i_3' \cup SP_1 \star (i_1'; i_2'; v_2') \cup SP_2 \star i_3' \\ & \quad \uplus SP \star Sig[SP \cup SP_1] \star (v_1'; i_2'; v_2') \cup SP_1 \star (i_1'; i_2'; v_2') \cup SP_2 \star i_3' \cup SP_1 \star (i_1'; i_2'; v_2') \cup SP_2 \star i_3' \\ & \quad \uplus SP_2 \star i_3' \cup SP_1 \star (i_1'; i_2'; v_2') \cup SP \star Sig[SP \cup SP_1] \star (v_1'; i_2'; v_2'). \end{aligned}$$

□

Note that Theorem 5.1 may be formulated more generally for any finite number of parameters, and in that case the proof would follow immediately from the current two parameters version by a simple induction.

The additional requirement that the result $SP(P_1 \Leftarrow v_1)$ of the first instantiation is chosen such that

$$Sig[SP_2] \cap Sig[SP(P_1 \Leftarrow v_1)] \subseteq Sig[SP \cup SP_1]$$

is essential in the proof of Theorem 5.1. If we disregarded this condition we may find situations where the isomorphism between the results of the two types of parameter instantiation may fail, as shown by the simple example below.

Example 5.8. Let us consider the following parameterized specification written in the CafeOBJ language.

```
mod* ELT { [ Elt ] }
mod* SP(E1 :: ELT, E2 :: ELT) {
  op a : -> Elt.E1
}
```

Given the fitting argument specifications

```
mod* SP1 {
  [ S ]
}
view v1 from ELT to SP1 { sort Elt -> S }

mod* SP2 {
  [ S ]
  op a : -> S
}
view v2 from ELT to SP2 { sort Elt -> S }
```

by the simultaneous instantiation SP(E1 + E2 ⇐ v1 + v2) we may obtain a specification with the following signature

```
[ S ]
ops a a' : -> S
```

If we instantiate SP sequentially we may choose a result of SP(E1 ⇐ v1) with the signature

```
[ Elt.E2 S ]
op a : -> S
```

which breaks the above hypothesis. Continuing with the instantiation SP(E1 ⇐ v1)(E2 ⇐ v2) we may get the signature

```
[ S ]
op a : -> S
```

which is *not* isomorphic with the one obtained through the simultaneous instantiation.

Note however that a proper choice of the instantiation $SP(P_1 \Leftarrow v_1)$ can always be made when the base institution strongly admits free extensions for idempotent-by-extension signature morphisms. In our benchmark example **MSA** this condition is guaranteed by the Proposition 5.1.

Corollary 5.3. *Let $SP(P_1, P_2)$ be a multiple parameterized specification and $v_i : P_i \to SP_i$, $i \in \{1,2\}$, two specification morphisms such that $Sig[P_i]$ and $Sig[SP_j]$ are disjoint, for $i, j \in \{1,2\}$. If in addition to the hypotheses of Proposition 5.4 the base institution strongly admits free extensions for idempotent-by-extension signature morphisms, then there exists an instantiation $SP(P_1 \Leftarrow v_1)$ (of the first parameter) such that any further instantiation $SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$ (of the second parameter) is isomorphic with the results $SP(P_1 + P_2 \Leftarrow v_1 + v_2)$ of the simultaneous instantiation, making the diagram below commutative.*

$$\begin{array}{ccc} SP(P_1 \cup P_2 \Leftarrow v_1 + v_2) & \xrightarrow{\cong} & SP(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \\ & \nwarrow^{\subseteq} \quad \nearrow_{\subseteq} & \\ & SP_1 \cup SP_2 & \end{array}$$

6. Conclusions

In this paper we have extended the set of the primitive institution-independent building operators for structuring specifications that is quite well established in the literature with new operators related to importation modes that are non-protecting and we have investigated new algebraic rules for the algebra of the model-oriented denotations of structured specifications determined by these building operators. Within the framework of our institution-independent specification structuring we have also extended the pushout-style parameterization concepts to the situation of sharing between the body of the parameterized module and the instance of the parameter, situation that corresponds to the actual realities of generic specification practice. Moreover, we have developed a set of abstract conditions naturally satisfied in the concrete specification frameworks that guarantee that the parallel and the serial instantiation of multiple parameters give isomorphic results. The checking of the conditions underlying this general result has been illustrated for the concrete case of structured specifications over many sorted algebra.

Our work leaves open a series of technical questions, such as to find sets of conditions naturally satisfied in the applications for upgrading the rules (19), (24), (29) from preorder to equivalence rules, and to extend the theory of parameterized specifications to situations that involve a higher level of sharing. For example we plan to consider sharing between different parameters, and between parameters and fitting argument specifications, situations that may occur quite naturally when the parameters use data types such as Booleans, numbers, etc.

Acknowledgements

The authors are grateful to both anonymous referees for very carefully studying their work, for finding a series of mistakes of various degrees, and for making a number of suggestions that have led to an improvement of this presentation.

Bibliography

[1] Jirí Adamek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories*. John Wiley, 1990.

[2] Marc Aiguier and Fabrice Barbier. An institution-independent proof of the Beth definability theorem. *Studia Logica*, 85(3):333–359, 2007.

[3] Jan Bergstra, Jan Heering, and Paul Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.

[4] Jan Bergstra and John Tucker. Elementary algebraic specifications of the rational complex numbers. In Kokichi Futatsugi, José Meseguer, and Jean-Pierre Jouannaud, editors, *Algebra, Meaning and Computation (a festschrift in honour of Professor Joseph Goguen)*, volume 4060 of *LNCS*, pages 459–475. Springer-Verlag Berlin Heidelberg, 2006.

[5] Francis Borceux. *Handbook of Categorical Algebra*. Cambridge University Press, 1994.

[6] Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286(2):197–245, 2002.

[7] Rod Burstall and Joseph Goguen. Putting theories together to make specifications. In Raj Reddy, editor, *Proceedings, Fifth International Joint Conference on Artificial Intelligence*, pages 1045–1058. Department of Computer Science, Carnegie-Mellon University, 1977.

[8] Rod Burstall and Joseph Goguen. The semantics of Clear, a specification language. In Dines Bjorner, editor, *1979 Copenhagen Winter School on Abstract Software Specification*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[9] Virgil Emil Căzănescu and Grigore Roşu. Weak inclusion systems. *Mathematical Structures in Computer Science*, 7(2):195–206, 1997.

[10] Răzvan Diaconescu. Elementary diagrams in institutions. *Journal of Logic and Computation*, 14(5):651–674, 2004.

[11] Răzvan Diaconescu. An institution-independent proof of Craig Interpolation Theorem. *Studia Logica*, 77(1):59–79, 2004.

[12] Răzvan Diaconescu. *Institution-independent Model Theory*. Birkhäuser, 2008.

[13] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.

[14] Răzvan Diaconescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge, 1993. Proceedings of a Workshop held in Edinburgh, Scotland, May 1991.

[15] Werner Fey. Pragmatics, concepts, syntax, semantics and correctness notions of ACT TWO: An algebraic module specification and interconnection language. Technical Report 88–26, Technical University of Berlin, Fachbereich Informatik, 1988.

[16] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[17] Joseph Goguen and Răzvan Diaconescu. An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(4):363–392, 1994.

[18] Joseph Goguen and José Meseguer. Order-sorted algebra solves the constructor selector, multiple representation and coercion problems. In *Proceedings, Second Symposium on Logic in Computer Science*, pages 18–29. IEEE Computer Society, 1987. Also Report CSLI-87-92, Center for the Study of Language and Information, Stanford University, March 1987; revised version in *Information and Computation, 103*, 1993.

[19] Joseph Goguen and Grigore Roşu. Composing hidden information modules over inclusive institutions. In *From Object-Orientation to Formal Methods*, volume 2635 of *Lecture Notes in Computer Science*, pages 96–123. Springer, 2004.

[20] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*. Kluwer, 2000.

[21] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

[22] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[23] José Meseguer. General logics. In H.-D. Ebbinghaus et al., editors, *Proceedings, Logic Colloquium, 1987*, pages 275–329. North-Holland, 1989.

[24] Peter D. Mosses, editor. *CASL Reference Manual*, volume 2960 of *Lecture Notes in Computer Science*. Springer, 2004.

[25] Andrei Popescu, Traian Şerbănuţă and Grigore Roşu. A semantic approach to interpolation. *Theoretical Computer Science*, 410(12-13):1109–1128, 2009.

[26] Grigore Roşu. Axiomatisability in inclusive equational logic. *Mathematical Structures in Computer Science*, 12(5):541–563, 2002.

[27] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Springer (in press).

[28] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Control*, 76:165–210, 1988.

[29] Andrzej Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37:269–304, 1986.

[30] Andrzej Tarlecki, Rod Burstall, and Joseph Goguen. Some fundamental algebraic tools for the semantics of computation, part 3: Indexed categories. *Theoretical Computer Science*, 91:239–264, 1991.