

Logical Support for Modularisation*

Răzvan Diaconescu Joseph Goguen[†]

Petros Stefaneas

Programming Research Group, Oxford University

Abstract: Modularisation is important for managing the complex structures that arise in large theorem proving problems, and in large software and/or hardware development projects. This paper studies some properties of logical systems that support the definition, combination, parameterisation and reuse of modules. Our results show some new connections among: (1) the preservation of various kinds of conservative extension under pushouts; (2) various distributive laws for information hiding over sums; and (3) (Craig style) interpolation properties. In addition, we study differences between syntactic and semantic formulations of conservative extension properties, and of distributive laws. A model theoretic property that we call exactness plays an important rôle in some results.

This paper explores the interplay between syntax and semantics, and thus lies in the tradition of abstract model theory. We represent logical systems as institutions. An important technical foundation is a new axiomatisation of the notion of inclusion. We also show how to subsume the deduction-based approach of π -institutions under that of ordinary institutions. Our results illuminate some interesting differences between equational style logics and first order style logics, encouraging us to conclude that, on the whole, equational style logics may be more suitable for modularisation than first order style logics.

1 Introduction

A promising approach to developing large and complex systems (which may be software, hardware, or both) is to start from a description of the system as an interconnection of some specification modules. This permits the verification of many properties to be carried out at the level of design, rather than code, and thus should improve reliability. With suitable mechanical support, it might also improve the efficiency of the development process. In addition, it promotes reuse, because some modules may be taken directly from a library, or else may be modifications of library modules. For this reason, many modern programming and specification languages support some form of modularisation, and most results about modules have appeared in the context of formal software engineering, particularly specification languages. Modularisation for theorem proving has been studied

*The research reported in this paper has been supported in part by grants from the Science and Engineering Research Council, and contracts with Fujitsu Laboratories Limited, and the Information Technology Promotion Agency, Japan, as part of the R & D of Basic Technology for Future Industries “New Models for Software Architecture” project sponsored by NEDO (New Energy and Industrial Technology Development Organization).

[†]Also, with SRI International, Menlo Park CA 94025.

less. This paper studies some properties of logical systems that support modularisation. We show that all reasonable institutions support certain simple operations on theories; we then explore what properties insure that these operations have various desirable properties. Our approach applies to theorem proving, software specification, hardware description, and logic-based programming; with some extra effort, it can even apply to imperative programming. Some general discussion and examples of our approach to modularisation are given in [?].

Two basic operations on theories are *sum* and *renaming*. The first simply combines the resources available from two or more theories, while the second changes the names of some internal features. Laws concerning such operations could be used, for example, to prove the correctness of an efficient implementation of a module interconnection facility in a programming environment. A third operation which makes this challenging is *information hiding*, whereby some internal features of a module are made inaccessible outside the module. It would be nice if everything true of the visible part of some combination of modules could be proved from what is known about the visible parts of the component modules. Unfortunately, this *Strong Distributive Law* often fails for real systems. However, we show that more restricted distributive laws hold for many important institutions. We also show that some other properties are more delicate, and only hold for institutions that are close to equational logic. Moreover, certain properties depend crucially on how they are formulated; for example, forms of Craig interpolation that are equivalent for institutions like first order logic are not at all equivalent for other important institutions.

A particularly important relation between theories is that of *conservative extension*, which says that any model of a subtheory can be expanded to a model of the supertheory. This semantic property can be important for the reuse of modules, and it implies a useful form of the distributive law. Other semantic properties of extensions arise in connection with parameterised (i.e., generic) modules.

A framework as abstract as institutions is needed for the comparative study of theories over logical systems. For example, an assertion of the form “the Craig interpolation property and the distributive law are equivalent” is not meaningful for a single logical system (such as first order logic); it needs to be quantified over logical systems. Perhaps an analogy will help to clarify this point. Suppose we are interested in properties of groups; two such properties might be “commutativity” and “finiteness.” Then asserting that “commutativity and finiteness are equivalent” for some group where both properties happen to hold (such as \mathbb{Z}_2) is valid, but not very meaningful; such an assertion should be universally quantified over groups (and then it is false).

Institutions are an abstraction of Tarski’s classic semantic definition of truth [63], based on a relation of satisfaction between models and sentences. In logic, there is a tradition, called *abstract model theory*, which abstracts the Tarskian approach to cover other logical systems (e.g., see [3, 2]). The goal of research in this area is to generalise as much of classical first order model theory as possible. Institutions can be seen in this light, but they are much more general, and are not focussed on first order logic. Institutions use category theory to achieve generality and simplicity; hence the concepts and arguments in this paper also need category theory. In particular, a new categorical axiomatisation of the notion of inclusion permits simple definitions for our operations on theories. A general discussion of how to apply ideas from category theory in Computing Science is given in [19].

One application for the machinery of this paper is theorem proving, where modules representing logical theories are important. In particular, these ideas are used in the 2OBJ theorem proving system [31], which supports deduction over any desired logical system by implementing its abstract data type of proofs in equational logic. 2OBJ builds on facilities from the OBJ3 system [?, 33], including its module system. The logical foundations of such an approach require formalising the notions of deduction in a logical system, and of encoding one logical system into another; these formalisations are provided by the notions of ruled charter (which gives rise to an institution) and ruled charter morphism; see [31], which builds on work in [22].

1.1 Some History

The earliest work on software modules with which we are familiar is by Parnas [47, 48, 49]. Program modules differ from earlier program structuring mechanisms such as subroutines, procedures and blocks, in that they may include a number of procedure and data definitions, may be parameterised, may import other modules, and may hide certain elements. A major motivation for modules in this sense is to facilitate the modification of software, by localising the representation of data and the operations that depend upon it; this is called *information hiding*. Such modules support software reuse because they can be specified, verified, and compiled separately. Note that this notion of module is essentially *syntactic*: it concerns texts that describe systems.

The earliest work that we know on specification modules is by Goguen and Burstall, for their specification language Clear [6, 7], the semantics of which is based on institutions. This approach to modules has been applied to various logic-based languages, particularly OBJ [?, ?] (an equational based on order sorted algebra), Eqlog [28] (which combines the functional and logic paradigms), FOOPS [29, 34] (which combines the functional and object paradigms), and FOOPlog [29] (which combines all three paradigms); it could also be applied to any pure logic-based programming languages, such as (pure) Lisp and (pure) Prolog. In [18], it is even extended to imperative programming.

Clear introduced the ideas that a specification module determines a theory, and that such theories can be put together using colimits; these ideas have their origin in some earlier work by Goguen on General Systems Theory [17, 24]. Clear provided operations for summing, renaming, extending, hiding, importing and (in the case of generics) applying theories. Theories in turn denote classes of models. The earliest work that we know giving a calculus of modules is also due to Goguen and Burstall [21]. Building on Clear, they studied laws for *horizontal* structuring relationships, and *vertical* implementing (also called “refinement”) relationships, concluding that the axioms of a 2-category should be satisfied. Some general laws for the module operations of Clear appear in [17], and others occur in the proofs in [7]. Some recent results on the formal properties of module composition over institutions appear in [20]. The present paper is in the same tradition.

The *module algebra* of Bergstra, Heering and Klint [4] attempts to capture the horizontal structure of modules with equations among certain basic operations on modules, including sum, renaming, and information hiding. These equations, together with constructors for signatures and sentences, give a many sorted equational presentation, about which some interesting results can be proved, including a normal form theorem. Unfortunately, this work has first order logic built into its choice of the constructors for signatures and sentences. However, Bergstra *et al.* abstract some interesting general principles from this

special case. In particular, we will see that the equivalence of Craig Interpolation with a distributive law for information hiding asserted in [4] is actually valid at the level of institutions under certain conditions (namely, compactness and closure under implication and false); moreover, we use essentially the same proof that they gave. Indeed, their paper was a major inspiration for the present paper, and opened what seems to us a fascinating realm of new questions in the theory of institutions.

The original semantics of Clear [7] did not capture the use of subsignatures and subtheories that is natural in many institutions. The present paper gives a new axiomatisation of the notion of inclusion, so that sums are given by least upper bounds of inclusions. A more concrete approach is given by Sannella and Tarlecki in [54]. The desirability of using inclusions for theories first appeared in the thesis of Sannella [53], which gave a set theoretic semantics for Clear in the case of many sorted equational logic.

Much interesting work using institutions has been done by Tarlecki [58, 59, 60, 61] and by Sannella and Tarlecki [54, 55, 56], and we discuss several aspects of this work later.

1.2 Relation with Type Theoretic Approaches

In systems like the Calculus of Constructions [11] and the various AUTOMATH languages [12], theories appear as dependent sequences of declarations, and are structured by various type theoretic devices. Hence, signatures and sentences are not cleanly separated; moreover, this proof-theoretic tradition has no model theory of the kind considered in this paper. Thus, it may seem unclear how to apply results in the present paper to work in this tradition.

The lack of model theory in type theoretic approaches is no problem, because Theorem 35 shows how to produce a model theory “out of thin air” for logical systems based on deduction, so that they can be seen as ordinary institutions. Alternatively, it should be possible to construct appropriate set theories for type theories like those of Martin-Löf, and then to construct appropriate models within these set theories.

Also, because our approach applies to theories before they are encoded, it should be possible to apply the machinery of this paper to theorem proving systems based on encodings into type theory, by structuring theories using our constructions, and then translating into type theory (which it should be possible to do automatically). However, we consider that research into this interesting area lies outside the scope of the present paper.

1.3 Summary of Results

This paper includes the following:

1. a novel categorical characterisation of the notion of “inclusion,” which is used to explicate the notion of “theory extension”;
2. a lifting of exactness from the signatures of an institution to its theories;
3. an embedding of the category of π -institutions into the category of ordinary institutions;
4. a general discussion of Craig interpolation for institutions with inclusions;

5. a discussion of the intuitive significance of different distributive laws for information hiding over the sum of theories;
6. a proof that Craig interpolation is equivalent to a certain distributive law, for reasonable compact institutions with implication and false;
7. results about how sets of sentences and sets of models behave with respect to sum and information hiding;
8. a study of distributive laws for sum over information hiding for classes of models;
9. a discussion of the relationship between the syntactic and semantic formulations of conservative extension;
10. a proof that the middle distributive law holds for conservative extensions in reasonable semiexact institutions;
11. a proof that a pushout of a conservative (or persistent) extension is again conservative (or persistent) in a reasonable semiexact institution; and
12. a proof that the pushouts of two extensions conservative for initiality are again conservative for initiality in a reasonable semiexact institution.

1.4 Prerequisites

This paper assumes familiarity with basic category theory, including categories, functors, limits, colimits, and adjoints. The necessary material may be found in [39], [38], or in a more gentle style, [35]. By way of notation, we use “;” for composition, we let 1_A denote the identity morphism at an object A , and we let $|\mathbf{C}|$ denote the class of objects of a category \mathbf{C} .

We have found that certain readers may have doubts about set theoretic foundations when category theory is used. In fact, this paper stays well away from anything that is problematical, and nearly any foundation that has been proposed for category theory will do, including the “hierarchy of universes” discussed e.g., by Mac Lane [39] in Section I.6.

1.5 Acknowledgements

We wish to thank Drs. José Fiadeiro, Piet Rodenburg, Don Sannella, and especially Andrzej Tarlecki, for their very useful comments on drafts of this paper. Prof. Virgil Emil Căzănescu carefully read several drafts and helped with some technical corrections. We also thank Prof. Rod Burstall for his important rôle in creating the theory of institutions. In addition, we thank Drs. José Meseguer, Donald Sannella, Andrzej Tarlecki, José Fiadeiro, Piet Rodenburg, and Jan Heering, as well as Profs. Thomas Maibaum, Jan Bergstra and Paul Klint for their inspiring papers, without which the present paper would not have been written.

2 Basic Concepts

This section presents some concepts that are needed for the rest of the paper. The first subsection summarises the necessary aspects of institutions, while the second discusses inclusions. The third subsection discusses exactness, and the fourth π -institutions.

2.1 Institutions

An adequate formalisation of logic as used in Computing Science must achieve a delicate balance between syntax and semantics. On the one hand, *syntax is fundamental*, because we deal with finite texts that (partially) describe VLSI chips, bank balances, computations, etc., and we manipulate such texts, e.g., with proofs, translations, and type checks. On the other hand, *semantics is fundamental*, because we are really interested in the *models*, not their descriptions; that is, we are interested in chips, computations, proofs, etc., and we manipulate the syntactic representations only because we hope that they correspond to reality¹.

Tarski’s semantic definition of truth for first order logic [63] is a traditional reconciliation of these two views of what is fundamental, based on the notion of *satisfaction* as a binary relation between models and sentences. Some such notion is needed for the very basic notions of *soundness* and *completeness* of logical systems, because these notions depend in an essential way upon the relationship between provability (which is syntactic) and satisfaction (which is semantic, i.e., concerns “truth” in Tarski’s sense). These notions, in turn, are basic to classical treatments of the adequacy of rules of deduction for logical systems; soundness and completeness with respect to an intuitively plausible class of models give us far greater confidence in a set of rules of deduction, and make their range of applicability more precise.

In a series of papers beginning in 1979, Burstall and Goguen developed institutions to formalise the intuitive notion of a logical system; the most recent and complete exposition is [23]. This approach allows us to discuss the crucial relationship between theories and models without commitment to either side at the expense of the other. Institutions are much more abstract than Tarski’s model theory, and they also add another basic ingredient, namely signatures and the possibility of translating sentences and models from one signature to another. A special case of this translation may be familiar from first order model theory: if $\Sigma \rightarrow \Sigma'$ is an inclusion of first order signatures, and if M is a Σ' -model, then we can form $M \upharpoonright_{\Sigma}$, called the *reduct* of M to Σ . Similarly, if e is a Σ -sentence, then we can always view it as a Σ' -sentence (but there is no standard notation for this). The key axiom, called the Satisfaction Condition, says that *truth is invariant under change of notation*, which is surely a very basic intuition for traditional logic.

Signatures are needed in Computing Science because we are less interested in “pure” logics than in “applied” logics, which have special “non-logical” symbols for the particularities of a given application area, such as VLSI chip design for fast Fourier transform. One of the major lessons of category theory is that it is not enough to consider just the objects, which in this case are signatures, but one should also consider the relevant “structure pre-

¹Actually, this is a rather naive view, because we never really “have reality,” but only models; further, we do not really “have models” either, but only descriptions of them. Moreover, these descriptions are often given using powerful idealistic constructions, such as power sets and dependent types, whose ultimate consistency cannot be proved.

“serving” morphisms, which in this case are signature morphisms. Without them, we would be unable to consider the variation of sentence, model, and satisfaction under change of notation.

Definition 1 An **institution** consists of

1. a category $Sign$, whose objects are called **signatures**,
2. a functor $Sen: Sign \rightarrow Set$, giving for each signature a set whose elements are called **sentences** over that signature,
3. a functor $Mod: Sign \rightarrow Cat^{op}$ 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 $\models_{\Sigma} \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$ for each $\Sigma \in |Sign|$, called Σ -**satisfaction**,

such that for each morphism $\phi: \Sigma \rightarrow \Sigma'$ in $Sign$, the **Satisfaction Condition**

$$M' \models_{\Sigma'} Sen(\phi)(e) \text{ iff } Mod(\phi)(M') \models_{\Sigma} e$$

holds for each $M' \in |Mod(\Sigma')|$ and $e \in Sen(\Sigma)$.

We may write $\phi(e)$ or even ϕe for $Sen(\phi)(e)$, and $M' \uparrow_{\phi}$ or possibly $\phi(M')$, for $Mod(\phi)(M')$; also we may drop the signature subscripts on the satisfaction relation when it is not confusing. If M is a Σ -model and E is a set of Σ -sentences, then we let $M \models_{\Sigma} E$ mean that $M \models_{\Sigma} e$ for each $e \in E$. Also, if E is a set of Σ -sentences and e is a single Σ -sentence, then we let $E \models_{\Sigma} e$ mean that for each Σ -model M , if $M \models_{\Sigma} E$ then $M \models_{\Sigma} e$. Finally, we let $\phi(E) = \{\phi(e) \mid e \in E\}$.

An institution is **compact** iff whenever $E \models e$ then there is a finite subset $E' \subseteq E$ such that $E' \models e$. \square

Of course, it takes some effort to prove that a given logical system really is an institution. The basic paper on institutions [23] showed that a number of basic logical systems are institutions, including the many sorted and unsorted versions of first order logic, equational logic, and Horn clause logic, with and without equality. Mosses shows that his unified algebras form an institution in [45]. Yukawa [67] gives an institution for the lambda calculus, with λ -models as models. Nielsen and Platet [46] give an institution for higher order logic with polymorphic types. Goguen and Burstall [22] show how to ease checking the Satisfaction Condition through the use of charters and parchments; this technique is applied by Stefanias [57] to get institutions for second order and modal logics.

Salibra and Scollo [52] propose weakening the notion of institution by eliminating the Satisfaction Condition, or replacing “iff” in it by “only if” or “if.” However, we feel that too much is lost by these generalisations. For example, the reader can check that many proofs in the present paper require both directions of the Satisfaction Condition. Moreover, we do not know any examples where this weakening is needed. Although [52] claims that hidden sorted equational logic is not an institution, but does satisfy half of the Satisfaction Condition, this is only because their version of hidden sorted equational logic has signature morphisms that do not respect all of the appropriate structure; see [20] for a proof that with appropriate signature morphisms, hidden sorted equational logic is indeed an institution.

Ehrig, Baldamus and Orejas [13, 14] have proposed a different weakening of institutions that they call “specification logics,” which are functors $Mod: Th^{op} \rightarrow Cat$ that assign a category of models to each (abstract) theory. It seems to us that this notion is rather far from that of logical system: there is no notion of satisfaction, nor even of sentence, and the invariance of truth under change of notation cannot be expressed. Moreover, many such functors have nothing to do with logic. Finally, we note that “specification logics” have earlier been studied under the name of (strict) indexed categories; for example, [62] proved a number of basic results about limits, colimits and the Grothendieck “flattening” construction. Thus, we believe that the name “specification logic” is misleading, although the concept is certainly useful. In our opinion, it is more interesting to explore variations of the institution concept that provide deduction, as well as satisfaction; see [23, 43] and [31] for further discussion.

There are two equivalent ways to give a denotation for a presentation P : the set of all sentences entailed by those in P ; and the class of all models of P . The first is the “closed presentation” or “theory” P^\bullet , and the second is the “closed model class” $\llbracket P \rrbracket$. Although one is syntactic and the other semantic, there is no essential difference between these two kinds of “closure” or “completion” of P ; there is a dual isomorphism because of the Galois connection of Proposition 3. However, significant differences do arise when one considers non-closed collections, because for the usual institutions, there are many more collections of models than there are collections of sentences. Indeed, usually there are only a countable number of finite sets of sentences, but there are so many different collections of models that one cannot even assign a cardinality; moreover, most of these collections could never arise in practice. The fact designers and verifiers often need to know about consequences of a specification that do not happen to be in its original presentation suggests using theories rather than just presentations. This leaves the practical problem of how to manipulate finite sets of sentences in a way that respects the behaviours of their (generally infinite) corresponding closed sets of sentences, and of course their corresponding models; in general, some form of theorem proving may be needed.

This paper does not give a formal definition for the word “module,” because different languages differ very widely in this respect. However, it may be useful to informally consider a module to be some kind of finite description for a presentation; then it is necessary to carefully distinguish such modules from their denotations. In Section 3.4, we discuss “module expressions” and their denotations.

It follows from our general discussion of syntax and semantics that we should define notions like “conservative extension” semantically, and then seek syntactic criteria for checking such relationships. Thus, we disagree with the approach of Maibaum, Sadler, and Turksi [40, 64], who consider only sentences and deduction. In our view, such an approach cannot adequately formalise the actual situation in (for example) software engineering, which involves models of concrete objects, as well as their descriptions. This is illustrated by the inability of the purely syntactic notion of conservative extension to capture the underlying semantic property, and by the fact that the syntactic notion is not preserved under pushouts for many important institutions. Similarly, we claim that operations on modules should have corresponding operations on their denotations, as opposed to merely being arbitrary textual “edit” operations; in this way, semantics provides a “reality check” for syntax.

We now give the precise definitions:

Definition 2 A **presentation** is a pair (Σ, E) where Σ is a signature and E is a set of Σ -sentences. Let E^* denote the collection of all Σ -models that satisfy each sentence in E . Given a collection M of Σ -models, let M^* denote the collection of all Σ -sentences that are satisfied by each model in M , and let M^* denote (Σ, M^*) , called the **theory** of M .

The **closure** of a collection E of Σ -sentences is E^{**} , denoted E^\bullet . A presentation (Σ, E) is **closed** iff $E = E^\bullet$, in which case it is called a **theory**. The Σ -theory **presented** by (Σ, E) is (Σ, E^\bullet) . A Σ -sentence e is **semantically entailed** by a collection E of Σ -sentences, written $E \models e$, iff $e \in E^\bullet$.

If $T = (\Sigma, E)$ is a theory, then its **denotation**, which we may write $\llbracket T \rrbracket$, is the full subcategory of $Mod(\Sigma)$ with objects M such that $M \models_\Sigma E$.

If (Σ, E) and (Σ', E') are presentations, then a **presentation morphism** $(\Sigma, E) \rightarrow (\Sigma', E')$ is a signature morphism $\phi: \Sigma \rightarrow \Sigma'$ such that $e \in E$ implies $\phi(e) \in E'$. Let $Pres(\mathcal{I})$ denote the category of presentations over an institution \mathcal{I} , and let $Th(\mathcal{I})$ denote its subcategory of theories over \mathcal{I} . \square

The following basic result is proved in [23].

Proposition 3 The two functions denoted “*” in Definition 2 form what is known as a **Galois connection** (see, e.g., [9]), in that they satisfy the following properties, for any collections E, E' of Σ -sentences and collections M, M' of Σ -models:

1. $E \subseteq E'$ implies $E'^* \subseteq E^*$.
2. $M \subseteq M'$ implies $M'^* \subseteq M^*$.
3. $E \subseteq E^{**}$.
4. $M \subseteq M^{**}$.

These imply the following properties:

5. $E^* = E^{***}$.
6. $M^* = M^{***}$.
7. There is a dual (i.e., inclusion reversing) isomorphism between the closed collections of sentences and the closed collections of models; this isomorphism takes unions to intersections and intersections to unions.

\square

The following basic result is proved in [23]:

Theorem 4 If the category of signatures of an institution \mathcal{I} has [finite] colimits, then so does its category $Th(\mathcal{I})$ of theories. \square

Again following [23], we now present institution morphisms, which are useful for comparing and for transferring results among different logical systems:

Definition 5 Let \mathcal{I} and \mathcal{I}' be institutions. Then an **institution morphism** $\Phi: \mathcal{I} \rightarrow \mathcal{I}'$ consists of

1. a functor $\Phi: \text{Sign} \rightarrow \text{Sign}'$,
2. a natural transformation $\alpha: \Phi; \text{Sen}' \Rightarrow \text{Sen}$, and
3. a natural transformation $\beta: \text{Mod} \Rightarrow \Phi; \text{Mod}'$

such that the following **Satisfaction Condition** holds

$$M \models_{\Sigma} \alpha_{\Sigma}(e') \text{ iff } \beta_{\Sigma}(M) \models'_{\Phi(\Sigma)} e'$$

for any Σ -model M from \mathcal{I} and any $\Phi(\Sigma)$ -sentence e' from \mathcal{I}' . Institutions with institution morphisms form a category, which we denote Ins . \square

The above is just one of several different kinds of morphism between institutions; while we feel that it is the one that properly reflects the structure of institutions (see [23] for detailed arguments on this point), it is not the right notion for certain purposes. For example, Salibra and Scollo [52] and Astesiano and Cerioli [1] describe some other possibilities.

2.2 Factorisations and Inclusions

In this subsection, we first present the standard notion of a factorisation system (e.g., see [38]), which has been used many places in Computing Science (e.g., [23, 58]); then we give our novel axiomatisation for the notion of “inclusion.”

Definition 6 An **image factorisation system** for a category \mathbb{C} consists of a class \mathcal{M} of monics and a class \mathcal{E} of epics in \mathbb{C} such that

- (1) both \mathcal{E} and \mathcal{M} are closed under composition,
- (2) all isomorphisms are in both \mathcal{M} and \mathcal{E} , and
- (3) every morphism f in \mathbb{C} can be factored as $e; m$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$ “uniquely up to isomorphism” in the sense that if $e'; m'$ is another factorisation of f with $e' \in \mathcal{E}$ and $m' \in \mathcal{M}$, then there is a unique isomorphism from the **centre object** C of $e; m$ (i.e., the target of e and the source of m) to the centre object C' of $e'; m'$ such that the diagram below commutes:

$$\begin{array}{ccc}
 & C & \\
 e \nearrow & & \searrow m \\
 A & & B \\
 e' \searrow & & \nearrow m' \\
 & C' &
 \end{array}$$

\square

The intuition for inclusion systems is that they capture “image subobjects” *uniquely*, rather than merely up to isomorphism. This not only better fits certain examples that interest us, but also has the technical advantage that subobjects are morphisms, instead of equivalence classes of morphisms, so that we can apply lattice operations to them directly.

Definition 7 An **inclusion system** for a category \mathbb{C} consists of a class \mathbb{I} of arrows and a class \mathcal{E} of epics in \mathbb{C} such that:

- (1) both \mathcal{E} and \mathbb{I} are subcategories of \mathbb{C} such that $|\mathcal{E}| = |\mathbb{I}| = |\mathbb{C}|$;
- (2) every morphism f in \mathbb{C} can be factored uniquely as $e; i$ with $e \in \mathcal{E}$ and $i \in \mathbb{I}$;
- (3) \mathbb{I} is a partial order (i.e., for any objects A, B , there is at most one morphism $A \rightarrow B$ in \mathbb{I} and if there is also a morphism $B \rightarrow A$ in \mathbb{I} , then $A = B$);
- (4) \mathbb{I} has finite least upper bounds (i.e., finite coproducts in \mathbb{I}), denoted $+$; and

We will call the morphisms in \mathbb{I} **inclusions**, and use the notation $A \hookrightarrow B$. Also, we let \emptyset denote the bottom element of \mathbb{I} (i.e., the empty least upper bound, which is also the initial object in \mathbb{I}). Given $i: A \hookrightarrow B$ and $f: B \rightarrow C$, we let $f \upharpoonright_A$ denote the composition $i; f$, and call it the **restriction** of f to A . We will also say that a category with an inclusion system is **inclusive**. \square

Note that neither inclusion systems nor factorisation systems are a special case of the other. Most of the categories used in specification and programming have obvious and natural inclusion systems, including the following:

Example 8 The category *Set* of sets with \mathbb{I} the inclusions and \mathcal{E} the surjections is an inclusion system. \square

Example 9 Recall (e.g., from [32]) that a many sorted signature Σ consists of a set S of sorts and an $S^* \times S$ -indexed family of sets $\Sigma_{w,s}$ whose elements are the operation symbols of arity (i.e., input sorts) $w \in S^*$, and value (i.e., output) $s \in S$. Also, a morphism $(S, \Sigma) \rightarrow (S', \Sigma')$ of many sorted signatures consists of a function $f: S \rightarrow S'$ and a $S^* \times S$ -indexed family of functions $g_{w,s}: \Sigma_{w,s} \rightarrow \Sigma'_{f(w),f(s)}$. Then we get an inclusion system in a natural way by defining \mathbb{I} by $(S, \Sigma) \hookrightarrow (S', \Sigma')$ iff $S \hookrightarrow S'$ and $\Sigma_{w,s} \hookrightarrow \Sigma'_{w,s}$ for each $w \in S^*$ and $s \in S$, and defining \mathcal{E} to consist of those $(f, g): (S, \Sigma) \rightarrow (S', \Sigma')$ such that $S' = f(S)$ and $\Sigma'_{w',s'} = \bigcup \{g(\Sigma_{w,s}) \mid f(w) = w' \text{ and } f(s) = s'\}$, for each w', s' . \square

Example 10 Similarly, the usual category of (unsorted) first order signatures has an obvious inclusion system. Recall that a first order signature is a pair (Σ, Π) , where both Σ and Π are ω -indexed families of sets, with Σ_n the operation symbols of arity n and Π_n the predicate symbols of arity n , for $n \in \omega$. Also, a morphism $(\Sigma, \Pi) \rightarrow (\Sigma', \Pi')$ is a pair (f, g) of ω -indexed functions, where $f_n: \Sigma_n \rightarrow \Sigma'_n$ translates operation symbols, and $g_n: \Pi_n \rightarrow \Pi'_n$ translates predicate symbols. Then we get an inclusion system by defining \mathbb{I} by $(\Sigma, \Pi) \hookrightarrow (\Sigma', \Pi')$ iff $\Sigma_n \hookrightarrow \Sigma'_n$ and $\Pi_n \hookrightarrow \Pi'_n$ for each $n \in \omega$, and defining \mathcal{E} to be the (component-wise) surjective signature morphisms. \square

Example 11 Any partial order with finite least upper bounds gives rise to a category \mathbb{I} that is an inclusion system with $\mathbb{C} = \mathbb{I}$ and $\mathcal{E} = |\mathbb{I}|$. \square

Although set theoretic inclusions are the simplest natural example of an inclusion system, inclusion systems in general may have properties that are quite different from those of sets. For example, any inclusion $A \hookrightarrow B$ in set theory is “split,” in the sense that B can be written as a disjoint union (i.e., coproduct) $A \cup C$ where C is the “difference” between B and A . This property does not hold for all inclusion systems; for example, consider the inclusion of many sorted signatures $(\{s\}, \phi) \hookrightarrow (\{s\}, \{f: s \rightarrow s\})$. However, the notion of inclusion system is sufficient to ensure that many familiar properties of inclusions hold, as shown in the rest of this subsection.

Proposition 12 If \mathbb{C} is a category with an inclusion system $(\mathbb{l}, \mathcal{E})$, then:

1. all inclusions are monics;
2. \mathbb{l} is **l-right cancellable**, in the sense that if $f;g \in \mathbb{l}$ and $g \in \mathbb{l}$, then $f \in \mathbb{l}$;
3. \mathcal{E} is **\mathcal{E} -left cancellable** (i.e., \mathcal{E}^{op} is \mathcal{E}^{op} -right cancellable);
4. all isomorphisms in \mathbb{l} are identities;
5. all coequalisers in \mathbb{C} are in \mathcal{E} ;
6. all isomorphisms in \mathbb{C} are in \mathcal{E} ;
7. any morphism f that is both in \mathcal{E} and \mathbb{l} is an identity; and
8. the category \mathbb{l} of inclusions has all finite colimits. If \mathbb{C} has an initial object \emptyset , then \emptyset is also initial in \mathbb{l} ; hence, \mathbb{C} has at most one initial object.

Proof: 1. Let $i \in \mathbb{l}$ and $u; i = v; i$. Consider $u = e; j$ and $v = e'; j'$ where $e, e' \in \mathcal{E}$ and $j, j' \in \mathbb{l}$. From the uniqueness of $(\mathbb{l}, \mathcal{E})$ -factorisations, we deduce that $e = e'$ and $j; i = j'; i$. But $j = j'$ because \mathbb{l} is a partial order. Therefore, $u = v$.

2. Factor f as $e; i$ with $e \in \mathcal{E}$ and $i \in \mathbb{l}$. Then $e; i; g$ is a factorisation of $f; g$, with $i; g \in \mathbb{l}$ because both i and g are inclusions. But $f; g$ also factors trivially as $1; (f; g)$, and so (2) of Definition 7 gives $e = 1$ and $f = i$. Therefore, $f \in \mathbb{l}$.

3. This is similar to 2.

4. By property (3) of Definition 7.

5. Let q be a coequaliser for u and v , i.e., an arbitrary parallel pair of arrows. Consider $q = e; i$ with $i \in \mathbb{l}$ and $e \in \mathcal{E}$. Then e is a coequaliser for u and v . Therefore, i is an isomorphism. Since any isomorphism in \mathbb{l} is an identity, we conclude that $q = e$.

6. Because any isomorphism is a coequaliser.

7. f can be factored in two different ways, as $f = 1; f$ and $f = f; 1$. Then $f = 1$ by property (3) of Definition 7.

8. \mathbb{l} trivially has coequalisers because it is a partial order, and it has finite coproducts by (4) of Definition 7. \square

Next, we discuss how the greatest lower and least upper bounds of objects in \mathbb{l} relate to constructions in \mathbb{C} ; these arguments have a set theoretic flavour.

Proposition 13 Let \mathbb{C} be a category with an inclusion system $(\mathbb{l}, \mathcal{E})$ and with pullbacks. Let $A \hookrightarrow B$ and $A' \hookrightarrow B$ be inclusions in \mathbb{C} . Then there is a unique pullback in \mathbb{C} of these inclusions such that the arrows of the pullback cone are also inclusions.

Proof: Let $(P \rightarrow A, P \rightarrow A')$ be a pullback in \mathbb{C} of the inclusions of A and A' into B . Now factor the pullback arrows as $P \rightarrow A = e; C \hookrightarrow A$ and $P \rightarrow A' = e'; C' \hookrightarrow A'$ where $e, e' \in \mathcal{E}$. Because of the uniqueness of the factorisation of $P \rightarrow B$, it follows that $C = C'$ and $e = e'$. It now follows that $(C \hookrightarrow A, C \hookrightarrow A')$ is also a pullback cone, and the uniqueness of C follows from 2. and 4. of Proposition 12. \square

Definition 14 Let \mathbb{C} be a category with an inclusion system $(\mathbb{l}, \mathcal{E})$ and with pullbacks. Let A and B be objects of \mathbb{C} . Then the **sum** of A and B , denoted $A + B$, is the coproduct of A and B in \mathbb{l} , and the **intersection** of A and B , denoted $A \cap B$, is the (unique) inclusion pullback in \mathbb{C} of the inclusions $(A \hookrightarrow A + B, B \hookrightarrow A + B)$ given by Proposition 13. \square

Proposition 15 Let \mathbb{C} be a category with an inclusion system $(\mathbb{l}, \mathcal{E})$ and with pullbacks. Then for any objects $A, B \in |\mathbb{C}|$ we have:

1. the sum $A + B$ is the least upper bound of A and B in \mathbb{l} ;
2. the intersection $A \cap B$ is the greatest lower bound of A and B in \mathbb{l} ;
3. sum and intersection are unique;
4. $A \cap B$ is included in A and B , and all three of these objects are included in $A + B$; and
5. the intersection is the pullback in \mathbb{C} of the sum.

Proof: The first two assertions follow directly from (4) of Definition 7. Then the sum $A + B$ is unique by its definition as the least upper bound in \mathbb{l} of A and B , and the intersection is unique because it is the greatest lower bound in \mathbb{l} .

The maps $A \cap B \hookrightarrow A$ and $A \cap B \hookrightarrow B$ are inclusions by Proposition 13, and thus $A \cap B \hookrightarrow A + B$ is a composite of inclusions.

That the intersection is the pullback in \mathbb{C} of the sum follows from the definition of intersections and the previous Proposition. \square

Definition 16 Let \mathbb{C} be a category with an inclusion system $(\mathbb{l}, \mathcal{E})$ and with pullbacks. Then the inclusion system $(\mathbb{l}, \mathcal{E})$ is **strong** iff for any objects $A, B \in |\mathbb{C}|$ their sum is the pushout of their intersection. \square

All examples previously discussed are strong inclusion systems. The property of the sum being the pushout of the intersection will play an important rôle in the rest of the paper.

Definition 17 An inclusion system \mathbb{l} for a category \mathbb{C} is **distributive** iff the following two laws hold for all objects A, B, C in \mathbb{C} :

1. $A + (B \cap C) = (A + B) \cap (A + C)$.
2. $A \cap (B + C) = (A \cap B) + (A \cap C)$.

\square

We can get a non-distributive inclusion system from any non-distributive lattice L , by letting \mathbb{C} be L regarded as the partial order category with an arrow $A \rightarrow B$ iff $A \leq B$ in L (see Example 11).

Definition 18 Let \mathbb{C} and \mathbb{C}' be categories with inclusion systems $(\mathbb{l}, \mathcal{E})$ and $(\mathbb{l}', \mathcal{E}')$, respectively. Then an **inclusion system morphism** is a functor $F: \mathbb{C} \rightarrow \mathbb{C}'$ such that $F \upharpoonright_{\mathbb{l}}: \mathbb{l} \rightarrow \mathbb{l}'$, and \cap is preserved. \square

The following expresses a basic relationship between pushouts and inclusions that is important in Section 5:

Definition 19 Let \mathbb{C} be an inclusive category. Then **pushouts preserve inclusions** in \mathbb{C} iff whenever a pair of arrows $(A \hookrightarrow A', A \rightarrow B)$ has a pushout, then they have a pushout of the form $(B \hookrightarrow B', A' \rightarrow B')$. \square

Definition 20 An institution with a [strong] inclusion system $(\mathbb{I}, \mathcal{E})$ on its category of signatures **respects** that inclusion system iff the functor $Sen : Sign \rightarrow Set$ is an inclusion system morphism, where Set has the inclusion system described in Example 8; for short, let us say that such an institution is [**strongly**] **inclusive**. Given $i : \Sigma \hookrightarrow \Sigma'$ and a Σ' -model M , we may write $i(M)$ as $M \upharpoonright_i$ or $M \upharpoonright_\Sigma$ and call it the **reduct** of M to Σ .

An inclusive institution is **distributive** iff the inclusion system on its signatures is distributive.

A presentation morphism $\varphi : (\Sigma, E) \rightarrow (\Sigma', E')$ over an inclusive institution is an **inclusion** iff φ is an inclusion; note that in this case, $E \subseteq E'^\bullet$. We also call such morphisms **extensions**. \square

Example 21 The many sorted algebraic institution is inclusive and distributive with the inclusion system of Example 9 on its signatures. \square

Example 22 The unsorted first order institution is inclusive and distributive with the inclusion system of Example 10 on its signatures. \square

Any non-distributive inclusion system can be turned into a trivial non-distributive inclusive institution, with the inclusion system as its category of signatures, and with no models and no sentences. However, all of the usual logical systems are distributive inclusive institutions, including first-order logic and equational logic.

Example 23 One might think that morphisms of inclusion systems should preserve the epi part of the inclusion system. That this cannot be assumed is shown by a counterexample for many sorted equational logic, communicated to us by Virgil Căzănescu. For this institution, the sentence functor, giving all many sorted equations for each signature, maps the surjective signature morphism $(\{s, s'\}, \{\sigma : s \rightarrow s', a : \rightarrow s'\}) \rightarrow (\{s\}, \{\sigma : s \rightarrow s, a : \rightarrow s\})$ to a non-surjective function between the corresponding sets of equations. For example, no term of the source signature translates to the term $\sigma(a)$ of the target signature. \square

Proposition 24 If \mathcal{I} is an institution with a factorisation system $(\mathbb{I}, \mathcal{E})$ on its signatures, then $(\mathbb{I}_{\mathbb{T}h}, \mathcal{E}_{\mathbb{T}h}) = (\{\Phi \in \mathbb{T}h \mid \Phi \in \mathbb{I}\}, \{\Phi : (\Sigma, E) \rightarrow (\Sigma', E') \in \mathbb{T}h \mid \Phi \in \mathcal{E} \text{ and } (\Phi(E))^\bullet = E'\})$ is a factorisation system on its theories. If \mathcal{I} respects an inclusion system $(\mathbb{I}, \mathcal{E})$ on its signatures, and if its category of signatures has finite colimits, then $(\mathbb{I}_{\mathbb{T}h}, \mathcal{E}_{\mathbb{T}h})$ is an inclusion system for $\mathbb{T}h(\mathcal{I})$.

Proof: Given a theory morphism $\Phi : (\Sigma, E) \rightarrow (\Sigma', E')$, let $\Phi = e; m$ be its factorisation in $Sign$, with centre object Σ'' . Let E'' be the closure of $e(E)$. Then $e : (\Sigma, E) \rightarrow (\Sigma'', E'')$ and $m : (\Sigma'', E'') \rightarrow (\Sigma', E')$ are theory morphisms, and so $e; m$ is a factorisation of Φ in $\mathbb{T}h$. It is easy to check that this forms a factorisation system.

If \mathcal{I} respects the inclusion system $(\mathbb{I}, \mathcal{E})$, then (2) and (3) of Definition 7 hold trivially, and $\mathbb{I}_{\mathbb{T}h}$ has finite coproducts by an argument similar to that given for Theorem 4 in [23]. \square

Fact 25 Under the hypothesis of Proposition 24, the forgetful functor $\mathbb{T}h(\mathcal{I}) \rightarrow \mathbf{Sign}$ is an inclusion system morphism. Moreover, it is limit and colimit preserving. \square

The assumptions encapsulated in the following play a basic rôle in this paper:

Definition 26 An institution is **reasonable** iff it is strongly inclusive and its signature category has finite colimits and finite limits. \square

All of the institutions that are widely used in Computing Science are reasonable, including classical first order logic and equational logic, in both sorted and unsorted variants.

Corollary 27 If (Σ, E) and (Σ', E') are theories over a reasonable institution, then

1. $(\Sigma, E) + (\Sigma', E') = (\Sigma + \Sigma', (E \cup E')^\bullet)$;
2. $(\Sigma, E) \cap (\Sigma', E') = (\Sigma \cap \Sigma', E \cap E')$.

Proof: The first equation follows from the proof of Theorem 4, and the same technique works for intersection, except that we do not need to take the closure because the intersection of the $(\Sigma \cap \Sigma')$ -theories $Sen(\Sigma) \cap E'$ and $Sen(\Sigma') \cap E$ is already a $(\Sigma \cap \Sigma')$ -theory. \square

Corollary 28 In any reasonable institution, the inclusion system on the category of theories is strong. \square

2.3 Exactness

An important model theoretic property of many logical systems is that finite colimits are preserved by the model functor. Thus, if we combine some theories T_i in a diagram $T: I \rightarrow \mathbb{T}h(\mathcal{I})$ having colimit (i.e., result of combination) C , then the denotations of the T_i and C behave in the way one would hope: $Mod(C)$ is the limit of the diagram T ; $Mod^{op}: I \rightarrow \mathbf{Cat}$. In particular (and assuming that the categories of Σ -models are concrete), our intuition would lead us to hope that a model of $T_1 \oplus T_2$ (the co-product) would consist of a pair of models, one of T_1 and the other of T_2 ; i.e., we intuitively expect $Mod(T_1 \oplus T_2)$ to be $Mod(T_1) \times Mod(T_2)$. The situation is similar for a pushout of theory morphisms $T_0 \rightarrow T_1$ and $T_0 \rightarrow T_2$, which for simplicity we assume are theory inclusions, so that T_0 is shared between T_1 and T_2 : we expect that a model of $T_1 \oplus_{T_0} T_2$ (the pushout) can be constructed from a pair of models, one of T_1 and the other of T_2 , by identifying their reducts to T_0 ; that is, we expect $Mod(T_1 \oplus_{T_0} T_2)$ to be the pullback of $Mod(T_1) \rightarrow Mod(T_0)$ and $Mod(T_2) \rightarrow Mod(T_0)$. This property, which we call *exactness*, seems to have first arisen in [56], and is also used in the pioneering work of Tarlecki [60] on abstract algebraic institutions, and of Meseguer [43] on categorical logics².

Definition 29 An institution is **exact** iff the model functor $Mod: \mathbf{Sign} \rightarrow \mathbf{Cat}^{op}$ preserves finite colimits, and is **semiexact** iff Mod preserves pushouts. \square

²Meseguer [43] introduced the term exactness, but used it for the concept that we call semiexactness here.

Although many sorted logics tend to be exact, their unsorted variants tend to be only semiexact. In particular, the model functor does not preserve coproducts for either unsorted first order logic or unsorted equational logic. This is undesirable from the point of view of modularisation. Combining this with the well known fact that the coproduct of unsorted terminating term rewriting systems need not be terminating, although it is terminating in the many sorted case, we might conclude that unsorted logics are unnatural for many applications in Computing Science.

It is not hard to see that any chartered institution is exact³. Charters were introduced by Goguen and Burstall [22] as a general way to produce institutions. The basic intuition is that the syntax of a logical system is an initial algebra. Because it appears that most institutions of interest in Computing Science can be chartered, it follows that most institutions of interest in Computing are exact. In particular, both many sorted first order logic and many sorted equational logic are exact. On the other hand, unsorted equational logic is not exact.

Notice that, for any institution \mathcal{I} , the model functor Mod extends to $Th(\mathcal{I})$, by mapping a theory (Σ, E) to the full subcategory $Mod(\Sigma, E)$ of $Mod(\Sigma)$ formed by the Σ -models that satisfy E . The following result shows that one can lift exactness from signatures to theories, so that exactness depends only on the behaviour of signatures, and is independent of what happens with sentences. Semiexactness for theories plays an important rôle in the “categorical logics” described by Meseguer in [43]. Here, we show that this follows from the corresponding property for signatures:

Proposition 30 If an institution is semiexact, then $Mod: Th \rightarrow Cat^{op}$ preserves pushouts.

Proof: Let $\phi_1: (\Sigma', E') \rightarrow (\Sigma_1, E_1)$ and $\phi_2: (\Sigma', E') \rightarrow (\Sigma_2, E_2)$ be morphisms of theories and let $\phi'_1: (\Sigma_2, E_2) \rightarrow (\Sigma, E)$ and $\phi'_2: (\Sigma_1, E_1) \rightarrow (\Sigma, E)$ be their pushout. Recall [23] that (ϕ'_1, ϕ'_2) is the pushout of (ϕ_1, ϕ_2) in $Sign$ and E is the closure of $\phi'_2(E_1) \cup \phi'_1(E_2)$.

Let M_1 be a Σ_1 -model of E_1 and M_2 a Σ_2 -model of E_2 such that $M_1 \upharpoonright_{\phi_1} = M_2 \upharpoonright_{\phi_2}$; now let M' denote this Σ' -model. Then by the Satisfaction Condition, M' satisfies E' . By semiexactness and the construction of pullbacks in Cat , there is a Σ -model M such that $M \upharpoonright_{\phi'_2} = M_1$ and $M \upharpoonright_{\phi'_1} = M_2$. By the Satisfaction Condition again, M satisfies the translations of both E_1 and E_2 , and thus satisfies E . We have now shown that any pair of models (M_1, M_2) with $M_1 \in |Mod(\Sigma_1, E_1)|$ and $M_2 \in |Mod(\Sigma_2, E_2)|$ and $M_1 \upharpoonright_{\phi_1} = M_2 \upharpoonright_{\phi_2}$ determines a (Σ, E) -model M .

Conversely, any (Σ, E) -model M is determined in this way by its translations $M_1 = M \upharpoonright_{\phi'_2}$ and $M_2 = M \upharpoonright_{\phi'_1}$ which, by the Satisfaction Condition, satisfy E_1 and E_2 , respectively.

Because the models of a theory form a full subcategory of the models of its signature, we can extend this argument to model morphisms. By semiexactness, $\upharpoonright_{\phi'_2}: Mod(\Sigma, E) \rightarrow Mod(\Sigma_1, E_1)$ and $\upharpoonright_{\phi'_1}: Mod(\Sigma, E) \rightarrow Mod(\Sigma_2, E_2)$ are the pullback of $\upharpoonright_{\phi_1}: Mod(\Sigma_1, E_1) \rightarrow Mod(\Sigma', E')$ and $\upharpoonright_{\phi_2}: Mod(\Sigma_2, E_2) \rightarrow Mod(\Sigma', E')$. \square

A proof of the following result was sketched in [56].

Corollary 31 If an institution is exact, then $Mod: Th \rightarrow Cat^{op}$ preserves finite colimits.

Proof: By exactness, Mod maps the initial object of $Sign$ to the terminal (singleton) category. Because the only model of this category satisfies the empty theory (i.e., the

³Using the facts that Mod is 2-representable for chartered institutions, and that 2-representable functors preserve colimits.

tautologies over the initial signature) we conclude that the model functor maps the initial theory to the terminal category. Now we are done, because all finite colimits can be constructed from pushouts and an initial object. \square

Here is another “lifting” result that will later help us with pushouts of conservative extensions:

Fact 32 In an inclusive institution, if pushouts preserve signature inclusions, then pushouts also preserve theory inclusions.

Proof: This follows from the fact that the forgetful functor $\mathbb{T}h(\mathcal{I}) \rightarrow \mathit{Sign}$ creates both colimits [23] and inclusions. \square

2.4 π -Institutions

It has been claimed (e.g., in [40]) that the standard notion of institution (Definition 1) is not suitable for the foundations of software engineering, because it is based on satisfaction rather than deduction. Following the work of Fiadeiro and Sernadas [16], logical systems based on deduction can be formalised as π -institutions, which have for each signature Σ a set of Σ -sentences, but no given models. To compensate for this lack, a consequence relation is given on sentences. We will use the definition of Fiadeiro and Sernadas [16] as modified by Meseguer [43], rather than that of Maibaum and Fiadeiro [41], which seems overly complex to us; Harper, Sannella and Tarlecki [37] have given a definition similar to Meseguer’s, but restricted to finite sets of sentences.

Meseguer [43] showed how to construct an institution from a system of deduction, by producing a model theory directly from a comma category construction on theories. We extend this construction to a functor, and use it to show that the category of π -institutions can be embedded into the category of institutions; hence, π -institutions can be seen as a special kind of ordinary institutions. Therefore the results of this paper apply can also be applied to deduction-based approaches to formal software engineering advocated by Turski and Maibaum [64] and Maibaum and Sadler [40], and later formalised by Fiadeiro and Maibaum using π -institutions [41].

Definition 33 A π -institution consists of

1. a category Sign whose objects are called **signatures**,
2. a functor $\mathit{Sen} : \mathit{Sign} \rightarrow \mathit{Set}$, giving for each signature a set whose elements are called **sentences** over that signature, and
3. a relation⁴ $\vdash_{\Sigma} \subseteq \mathcal{P}(\mathit{Sen}(\Sigma)) \times \mathit{Sen}(\Sigma)$ for each $\Sigma \in |\mathit{Sign}|$, called Σ -**consequence**,

such that the following conditions hold:

- A. **reflexivity:** $\{e\} \vdash_{\Sigma} e$ for each $e \in \mathit{Sen}(\Sigma)$;
- B. **monotonicity:** if $E \vdash_{\Sigma} e$ and $E \subseteq E'$ then $E' \vdash_{\Sigma} e$;
- C. **transitivity:** if $E \vdash_{\Sigma} e'$ for each $e' \in E'$ and if $(E \cup E') \vdash_{\Sigma} e$, then $E \vdash_{\Sigma} e$;

⁴Here \mathcal{P} denotes the power set function.

D. translation: if $E \vdash_{\Sigma} e$ and if $\phi: \Sigma \rightarrow \Sigma'$ in $Sign$, then $\phi(E) \vdash_{\Sigma'} \phi(e)$.

A π -institution is **compact** iff whenever $E \vdash_{\Sigma} e$ then there is some finite $E' \subseteq E$ such that $E' \vdash_{\Sigma} e$. \square

Notice that although π -institutions axiomatise a syntactic consequence relation, they do not axiomatise an underlying notion of deduction.

Definition 34 Let \mathcal{I} and \mathcal{I}' be π -institutions. Then a π -**institution morphism** $\Phi: \mathcal{I} \rightarrow \mathcal{I}'$ consists of

1. a functor $\Phi: Sign \rightarrow Sign'$, and
2. a natural transformation $\alpha: \Phi; Sen' \Rightarrow Sen$,

such that $E \vdash_{\Phi(\Sigma)} e$ implies $\alpha_{\Sigma}(E) \vdash_{\Sigma} \alpha_{\Sigma}(e)$ for any signature Σ , for any set E of $\Phi(\Sigma)$ -sentences, and any $\Phi(\Sigma)$ -sentence e . Let $\pi\text{-Ins}$ denote the category of π -institutions with π -institution morphisms. \square

There is a forgetful functor $\mathcal{U}: Ins \rightarrow \pi\text{-Ins}$ which maps any institution to a corresponding deductive system, by forgetting the models, and by letting $E \vdash e$ mean $E \models e$. Applying \mathcal{U} to an institution morphism forgets its model part, and the Satisfaction Condition for institution morphisms helps to prove that this indeed gives a π -institution morphism.

In [43], Meseguer gives a comma category construction for categories of models of a π -institution. It is remarkable that this construction gives exactly the right models for the most common examples. The following result extends this construction to institution morphisms, thus giving a functor that embeds the category of π -institutions as a retract into the category of institutions.

Theorem 35 The forgetful functor $\mathcal{U}: Ins \rightarrow \pi\text{-Ins}$ has a retract (i.e., a left inverse functor) $\mathcal{F}: \pi\text{-Ins} \rightarrow Ins$.

Proof: The functor \mathcal{F} maps a π -institution $(Sign, Sen, \vdash)$ to the institution $(Sign, Sen, Mod, \models)$ with the same category of signatures and the same sentence functor. Given a signature Σ , the category $Mod(\Sigma)$ is defined to be the comma category $(\Sigma, \emptyset^{\bullet})/Th$. Thus, a Σ -model is just a theory morphism $G_1: (\Sigma, \emptyset^{\bullet}) \rightarrow (\Sigma_1, E_1)$, and a model morphism $H: G_1 \rightarrow G_2$ is just a theory morphism $(\Sigma_1, E_1) \rightarrow (\Sigma_2, E_2)$ such that $G_2 = G_1; H$.

If $D: \Sigma' \rightarrow \Sigma$ is a signature morphism, then a Σ -model G is translated to the Σ' -model $D; G$. The satisfaction of a Σ -sentence e by any Σ -model $G_1: (\Sigma, \emptyset^{\bullet}) \rightarrow (\Sigma_1, E_1)$ is defined by $G_1(e) \in E_1$. It is easy to check that these data define an institution.

Now consider a π -institution morphism $(\Phi, \alpha): (Sign, Sen, \vdash) \rightarrow (Sign', Sen', \vdash')$. We let \mathcal{F} map this to $(\Phi, \alpha, \beta): (Sign, Sen, Mod, \models) \rightarrow (Sign', Sen', Mod', \models')$, where only the natural transformation $\beta: Mod \Rightarrow \Phi^{op}; Mod'$ has still to be defined. For any signature Σ , β_{Σ} maps the model $G_1: (\Sigma, \emptyset^{\bullet}) \rightarrow (\Sigma_1, E_1)$ to $\Phi(G_1): (\Phi(\Sigma), \emptyset^{\bullet}) \rightarrow (\Phi(\Sigma_1), \alpha_{\Sigma_1}^{-1}(E_1))$. That $\alpha_{\Sigma_1}^{-1}(E_1)$ is a theory follows from the Satisfaction Condition on α . Also, a model morphism $H: G_1 \rightarrow G_2$ is mapped to the model morphism $\Phi(H): \Phi(G_1) \rightarrow \Phi(G_2)$, and $\Phi(H)$ is indeed a morphism of theories $(\Phi(\Sigma_1), \alpha_{\Sigma_1}^{-1}(E_1)) \rightarrow (\Phi(\Sigma_2), \alpha_{\Sigma_2}^{-1}(E_2))$, by the naturality of α .

To show the Satisfaction Condition for institution morphisms, we have to check that for any Σ -model $G_1 : (\Sigma, \emptyset^\bullet) \rightarrow (\Sigma_1, E_1)$ and any $\Phi(\Sigma)$ -sentence e' , we have $G_1 \models_\Sigma \alpha_\Sigma(e')$ iff $\beta_\Sigma(G_1) \models'_{\Phi(\Sigma)} e'$. This condition is equivalent to $G_1(\alpha_\Sigma e') \in E_1$ iff $\Phi(G_1)e' \in \alpha_{\Sigma_1}^{-1}(E_1)$. This holds because $\alpha_\Sigma; G_1 = \Phi(G_1); \alpha_{\Sigma_1}$ by the naturality of α .

This completes the definition of \mathcal{F} , and it is not hard to check that $\mathcal{F};\mathcal{U}$ is the identity. \square

Thus, given a π -institution \mathcal{P} , we get an ordinary institution $\mathcal{F}(\mathcal{P})$. It should be clear what it means for \mathcal{P} to be inclusive, distributive, etc., and it is easy to see that these properties carry over to $\mathcal{F}(\mathcal{P})$, because they only depend on *Sign*. Because $\mathcal{P} = \mathcal{U}(\mathcal{F}(\mathcal{P}))$ and because \mathcal{U} interprets \vdash as \models , the theories of \mathcal{P} are exactly the same as the theories of $\mathcal{F}(\mathcal{P})$. Also, Meseguer [43] showed that $\mathcal{F}(\mathcal{P})$ is semiexact. Putting these results together, we are able to apply results about institutions to $\mathcal{F}(\mathcal{P})$, and hence to \mathcal{P} .

3 Basic Module Algebra

Bergstra, Heering and Klint [4] consider a *module* to be a set of sentences in first order logic, and develop a “module algebra” which captures many important properties of modules in this sense, through equations that are satisfied by basic operations on modules. This notion of module is appropriate for some applications to specification languages, and in any case, it is interesting to know what equations are satisfied. However, we wish to note that this approach does not capture generic modules, nor can it characterise the relationships of import and export that may hold between two modules. Also, we see no way that such an approach can capture the various notions of conservative extension that are treated in this paper. Thus, some important aspects of modules are not “algebraic” in this sense.

This section and the next discuss some equations that may be satisfied by operations on theories over a reasonable institution \mathcal{I} . Recall that we have already shown that $\mathbb{T}h(\mathcal{I})$ has an inclusion system and finite colimits when \mathcal{I} is reasonable. We will use the following notations (from [4]): if $T = (\Sigma, E)$, then we let $\Sigma(T) = \Sigma$; and given a signature Σ , we let $T(\Sigma) = (\Sigma, \emptyset^\bullet)$ and we call it the **empty theory**. Notice that even for equational logic, there may be some sentences in $T(\Sigma)$, for example, those of the form $(\forall x) x = x$.

3.1 Sum

A very basic operation on theories is simply to *combine* their features into a single theory. This was modelled with colimits in Clear [7] in a way that permitted shared subtheories. For a strongly inclusive institution, we can use the specific colimit that is given by the sum.

Recall that given theories T and T' , their sum is their coproduct $T + T'$ in $\mathbb{T}h$. In examples from concrete institutions such as many sorted equational logic and first order logic, $T + T'$ is in general not a disjoint union, because there may be some sharing between T and T' .

Proposition 36 Given theories T, T', T'' , then:

1. $T + T' = T' + T$.
2. $T + (T' + T'') = (T + T') + T''$.

$$3. T + \emptyset = T.$$

$$4. T + T = T.$$

Proof: These are immediate from Proposition 24. \square

3.2 Renaming

In the practical development of large proofs, or large hardware and/or software systems, it can be very helpful to *reuse* texts. However, sometimes the source texts will not have the right names for the intended reuse; for example, we may wish to avoid having some variables be shared between two given modules. In such circumstances, it is important to be able to *rename* features. The operation \star described below accomplishes this. (Recall that we are assuming a reasonable institution.)

Definition 37 Let $\phi: \Sigma \rightarrow \Sigma'$ be a signature morphism and let $T = (\Sigma, E)$ be a Σ -theory. Then we let $\phi \star T = (\Sigma', (\phi(E))^\bullet)$. \square

Fact 38 Let $\phi: \Sigma \hookrightarrow \Sigma'$ be an inclusion and let $T = (\Sigma, E)$ be a Σ -theory. Then $\phi \star T = (\Sigma', E^\bullet)$.

Proof: Because $Sen(\phi): Sen(\Sigma) \rightarrow Sen(\Sigma')$ is an inclusion, we have that $\phi(e) = e$ for all $e \in Sen(\Sigma)$, and hence $\phi(E) = E$. \square

Proposition 39 Let T and T' be theories, and let ϕ and ψ be signature morphisms. Then:

1. If $\phi: \Sigma(T) \rightarrow \Sigma'$ and $\psi: \Sigma' \rightarrow \Sigma''$, then $\psi \star (\phi \star T) = (\phi; \psi) \star T$.
2. If $i: \Sigma(T) \hookrightarrow \Sigma''$, $i': \Sigma(T') \hookrightarrow \Sigma''$, $j: \Sigma(T) + \Sigma(T') \hookrightarrow \Sigma''$ and $\phi: \Sigma'' \rightarrow \Sigma'''$, then $j; \phi \star (T + T') = (i; \phi \star T) + (i'; \phi \star T')$. (One might write this loosely as $\phi \star (T + T') = (\phi \star T) + (\phi \star T')$.)

Proof: First, let T be (Σ, E) and T' be (Σ', E') . Then

$$\psi \star (\phi \star T) = \psi \star (\Sigma', (\phi(E))^\bullet) = (\Sigma'', \psi((\phi(E))^\bullet)^\bullet) = (\Sigma'', (\psi(\phi(E))^\bullet)^\bullet) = (\phi; \psi) \star T,$$

where the second step follows by noting that the Satisfaction Condition implies that any consequence of $\phi(E)$ is mapped by ψ into a consequence of $\psi(\phi(E))$.

For the second assertion, we will rewrite both sides of the equation to the same expression:

$$j; \phi \star (T + T') = (\Sigma''', (\phi(E \cup E'))^\bullet)^\bullet = (\Sigma''', (\phi(E \cup E'))^\bullet)$$

and

$$\begin{aligned} (i; \phi \star T) + (i'; \phi \star T') &= (\Sigma''', (\phi E)^\bullet)^\bullet + (\Sigma''', (\phi E')^\bullet)^\bullet = (\Sigma''', ((\phi E)^\bullet \cup (\phi E')^\bullet)^\bullet) \\ &= (\Sigma''', (\phi(E \cup E'))^\bullet)^\bullet. \end{aligned}$$

\square

One way to implement a module system with renaming is to consider that the “real name” of an operation is qualified by the name of the module where it is defined; then one can use short names when there are no clashes, and one can avoid clashes by using the long names if necessary. This solution has been implemented in the OBJ3 system [?] simply by renaming all operations as they are stored into the module database. Then module importation is just inclusion of the renamed theories. Thus, our mathematical treatment of theory inclusion already takes account of this practical difficulty.

3.3 Information Hiding

Information hiding is an important technique in modern programming, as well as in algebraic specification. Parnas [47] emphasised the importance of hiding implementation details within a module, in order to make it possible (for example) to improve a given data representation without having to search through all of a large program for each place where details of the representation are used. This is accomplished by *hiding* the data representation, i.e., by allowing access to it only through operations exported by its module. Similarly, Majster [42] showed that certain Σ -algebras cannot be specified as the initial Σ -algebra of a finite set of Σ -equations, while later work by Bergstra and Tucker (see [5], and a summary of related research in [44]) showed that any recursive Σ -algebra could be specified as the Σ -restriction of an initial Σ' -algebra of a finite set of Σ' -equations. Thus, there are some interesting Σ -theories that do not have finite Σ -presentations, but that are Σ -restrictions of finitely presented Σ' -theories, for some $\Sigma \subseteq \Sigma'$. In both cases, we have a signature inclusion $\Sigma \hookrightarrow \Sigma'$, and a Σ -theory $\Sigma \square T$ of what is visible, derived by restricting a Σ' -theory T that includes both the visible and the hidden features. (Recall that we are assuming a reasonable institution.)

Definition 40 Let Σ' be a signature and let $T = (\Sigma, E)$ be a theory. Then

$$\Sigma' \square T = (\Sigma \cap \Sigma', E \cap \text{Sen}(\Sigma')),$$

where \square is called the **information hiding operator**. \square

Bergstra, Heering and Klint [4] called \square the “export operator,” but we think that this name may be misleading.

Proposition 41 Given a signature inclusion $\Sigma' \hookrightarrow \Sigma$ and a Σ -theory T , then $\Sigma' \square T$ is a Σ' -theory.

Proof: Let e be a Σ' -sentence such that $E \cap \text{Sen}(\Sigma') \models_{\Sigma'} e$. Then we have to prove that $e \in E \cap \text{Sen}(\Sigma')$. To prove that $e \in E$, it suffices to prove that $E \models_{\Sigma} e$. Let M be a model of E . Then also $M \models_{\Sigma} E \cap \text{Sen}(\Sigma')$. By the Satisfaction Condition, the reduct $M \upharpoonright_{\Sigma'}$ also satisfies $E \cap \text{Sen}(\Sigma')$. It follows that $M \upharpoonright_{\Sigma'} \models_{\Sigma'} e$. Now going backwards with the Satisfaction Condition, we obtain that $M \models_{\Sigma} e$, which proves this result. \square

Corollary 42 For any signature Σ' and any Σ -theory T , $\Sigma' \square T$ is a $(\Sigma \cap \Sigma')$ -theory. \square

3.4 Module Expressions

Modules are defined in a wide variety of ways in programming, specification and theorem proving languages. For example, Clear modules are built up recursively from its combine, enrich, derive, and apply operations. The module expressions of the OBJ language are generated in a similar way. Standard ML [36] and Modula-2 [66] also have interesting module systems. In the context of the present section, we may define module expressions to be built up from $+$, \square , and \star , plus the finite presentations; i.e., they are the elements of the free term algebra⁵. Much of this paper can be seen as exploring equations that hold on the quotient of this algebra by an equivalence induced by some notion of denotation. What is attractive about the module algebra of Bergstra *et al.* [4] is that it provides a normal form for such module expressions; unfortunately, their development is restricted to the case of first order logic.

4 Distributive Laws

This section discusses three distributive laws for information hiding over the sum of theories. The strong distributive law is what we really wish were true, because it says that to prove something about the visible part of any theory, we do not need to know anything about what is invisible; this would greatly simplify reasoning about theories with hidden parts. However, this law does not seem to be satisfied by any institution that is widely used in Computing Science. The middle distributive law seems to be about the strongest law that does hold in some institutions of interest; it is a restriction of the strong law by some conditions on the signatures involved. The weak distributive law is so weak that it is satisfied by almost all institutions that are widely used in Computing Science.

Definition 43 Let \mathcal{I} be a reasonable institution. Then \mathcal{I} satisfies the **strong distributive law** iff given a Σ_1 -theory T_1 , a Σ_2 -theory T_2 , and a signature Σ' such that $\Sigma' \hookrightarrow \Sigma_1$ and $\Sigma' \hookrightarrow \Sigma_2$, then

$$\Sigma' \square (T_1 + T_2) = (\Sigma' \square T_1) + (\Sigma' \square T_2).$$

\mathcal{I} satisfies the **(middle) distributive law** iff given a Σ_1 -theory T_1 and a Σ_2 -theory T_2 , then

$$(\Sigma_1 \cap \Sigma_2) \square (T_1 + T_2) = ((\Sigma_1 \cap \Sigma_2) \square T_1) + ((\Sigma_1 \cap \Sigma_2) \square T_2).$$

\mathcal{I} satisfies the **weak distributive law** iff given a Σ_1 -theory T_1 and signatures Σ_2 and Σ' such that $\Sigma' \hookrightarrow \Sigma_1$ and $\Sigma' \hookrightarrow \Sigma_2$, then

$$\Sigma' \square (T_1 + T(\Sigma_2)) = (\Sigma' \square T_1) + (\Sigma' \square T(\Sigma_2)).$$

□

Notice that a reasonable institution satisfies the (middle) distributive law iff it satisfies the strong distributive law in the particular case where Σ' is $\Sigma(T_1) \cap \Sigma(T_2)$. Also, a reasonable institution satisfies the weak distributive law iff it satisfies the strong distributive law in the special case where one of the theories is the empty theory.

⁵A slightly subtle point is that a sort constraint is needed for application of the renaming operation.

The following counterexample shows that equational logic does not satisfy the strong distributive law. Actually, this example applies to any reasonable institution that supports an internalisation for equality and for constants. This includes not only first order logic with equality, but also ordinary first order logic, by introducing a binary relational symbol for equality plus sentences expressing its transitivity and symmetry.

Example 44 Let Σ' be an unsorted signature with just the two constants 1 and 2, and let Σ_1 and Σ_2 be Σ' with an additional constant 0. Let T_1 be the Σ_1 -theory generated by the equation $1 = 0$, and let T_2 be the Σ_2 -theory generated by the equation $2 = 0$. Then $\Sigma' \sqcap T_1$ and $\Sigma' \sqcap T_2$ are both empty, and so their sum is also empty. But $T_1 + T_2$ contains the equation $1 = 2$, which is also contained in $\Sigma' \sqcap (T_1 + T_2)$ because it is a Σ' -sentence. \square

4.1 Distributive Laws and Interpolation

There are many different formulations of the Craig Interpolation Property at the level of institutions. Some that are equivalent for first order logic differ for many other important institutions, and some are so strong that they are not satisfied even by first order logic. We first formulate the Craig Interpolation Property for institutions in a style used by Rodenburg [51] for the equational case. Then we connect the middle distributive law with Craig Interpolation, in the style of [4] and [50].

Definition 45 A reasonable institution satisfies the **Craig Interpolation Property** iff for any signatures Σ and Σ' and any set E of Σ -sentences and any set E' of Σ' -sentences, if $E \models_{\Sigma+\Sigma'} E'$ then there is a set I of $(\Sigma \cap \Sigma')$ -sentences such that $E \models_{\Sigma} I$ and $I \models_{\Sigma'} E'$. The set I is called the **interpolant** of E and E' . \square

Without axiomatising inclusions, it is not possible to formulate Craig Interpolation in quite such an intuitive way. For example, Tarlecki [59] gives a more restrictive definition, involving a pushout of arbitrary signature morphisms. Another difference is that we consider *sets* of sentences rather than just single sentences. We agree with Rodenburg [51] that this is more natural; in particular, note that equational logic satisfies Definition 45, but not the single sentence version given in [59]. In an institution with (arbitrary) conjunctions, the single sentence and the set of sentence forms are equivalent; but equational logic does not have even finite conjunctions.

Maibaum and Sadler [40] give a version of Craig Interpolation that is stronger in a different way, and Maibaum and Fiadeiro [41] give a precise formulation of it for π -institutions without inclusions. The following formulates this notion for inclusive institutions:

Definition 46 A reasonable institution \mathcal{I} has the **Strong Craig Interpolation Property** iff for any theories (Σ_1, E_1) , (Σ_2, E_2) and any Σ_2 -sentence e , if $E_1 \cup E_2 \models_{\Sigma_1+\Sigma_2} e$, then there is a set I of $(\Sigma_1 \cap \Sigma_2)$ -sentences such that $E_1 \models_{\Sigma_1} I$ and $E_2 \cup I \models_{\Sigma_2} e$. \square

In order to state the relationships among Craig Interpolation, Strong Craig Interpolation, and the distributive law, we need the following:

Definition 47 An institution is **closed under implication** iff for any signature Σ , for any finite set E of Σ -sentences and any single Σ -sentence e , there is a set E' of Σ -sentences such that a Σ -model M satisfies E' iff it satisfies e whenever it satisfies E . We may write $(E \Rightarrow e)$ for this set E' . An institution is **false** iff for any signature Σ , there is a Σ -sentence $false_{\Sigma}$ such that no Σ -model satisfies it. \square

There are several ways to formulate results like the following, depending for example on whether closure under finite or arbitrary implication is assumed; but all of the proofs would be similar to that given below.

Proposition 48 If a reasonable \mathcal{I} is compact and closed under implication, then it satisfies Craig Interpolation iff it satisfies Strong Craig Interpolation.

Proof: Craig Interpolation is the special case of Strong Craig Interpolation with $E_2 = \emptyset$, so we only have to show that a compact institution with implication and Craig Interpolation satisfies Strong Craig Interpolation. Thus, assume that we are given theories (Σ_1, E_1) and (Σ_2, E_2) , and let e be a Σ_2 -sentence such that $E_1 \cup E_2 \models_{\Sigma} e$, here $\Sigma = \Sigma_1 + \Sigma_2$. By compactness, there are finite sets $E'_1 \subseteq E_1$ and $E'_2 \subseteq E_2$ such that $E'_1 \cup E'_2 \models_{\Sigma} e$, and by closure under implication, there is a set $(E'_2 \Rightarrow e)$ of Σ_2 -sentences such that for any Σ_2 -model M , we have $M \models_{\Sigma_2} E'_2$ implies $M \models_{\Sigma_2} e$ iff $M \models_{\Sigma_2} (E'_2 \Rightarrow e)$. Then $E'_1 \models_{\Sigma} (E'_2 \Rightarrow e)$, because $E'_1 \cup E'_2 \models_{\Sigma} e$, by applying the Satisfaction Condition for the inclusion $\Sigma_2 \hookrightarrow \Sigma$. Craig Interpolation now gives us a set I of $\Sigma_1 \cap \Sigma_2$ -sentences such that $E'_1 \models_{\Sigma_1} I$ and $I \models_{\Sigma_2} (E'_2 \Rightarrow e)$. Thus, $E'_1 \models_{\Sigma_1} I$ and $I \cup E'_2 \models_{\Sigma_2} e$, which implies that $E_1 \models_{\Sigma_1} I$ and $I \cup E_2 \models_{\Sigma_2} e$. \square

Theorem 49 Let \mathcal{I} be a reasonable compact institution closed under implication. Then \mathcal{I} satisfies the distributive law if it satisfies the Craig Interpolation Property. Moreover, if \mathcal{I} is a reasonable compact institution closed under implication and false, then it satisfies the distributive law iff it satisfies the Craig Interpolation Property.

Proof: Let $T_1 = (\Sigma_1, E_1)$ and $T_2 = (\Sigma_2, E_2)$ be theories. Let Σ' be $\Sigma(T_1) \cap \Sigma(T_2)$. We now calculate the two sides of distributive law. First,

$$\Sigma' \square (T_1 + T_2) = \Sigma' \square (\Sigma_1 + \Sigma_2, (E_1 \cup E_2)^{\bullet}) = (\Sigma', \text{Sen}(\Sigma') \cap (E_1 \cup E_2)^{\bullet}),$$

which is a theory by Proposition 41. Next,

$$(\Sigma' \square T_1) + (\Sigma' \square T_2) = (\Sigma', E_1 \cap \text{Sen}(\Sigma')) + (\Sigma', E_2 \cap \text{Sen}(\Sigma')) = (\Sigma', ((E_1 \cup E_2) \cap \text{Sen}(\Sigma'))^{\bullet}).$$

Now let E be the set of $(\Sigma_1 + \Sigma_2)$ -sentences which is the set theoretic union of E_1 and E_2 . Notice that because $\text{Sen}(\Sigma') \cap E^{\bullet}$ is a theory (by Proposition 41 again) and because $E \cap \text{Sen}(\Sigma')$ is included in it, we have $(E \cap \text{Sen}(\Sigma'))^{\bullet} \subseteq \text{Sen}(\Sigma') \cap E^{\bullet}$. We will show that the opposite inclusion follows from the Craig Interpolation Property when \mathcal{I} is compact and closed under implication.

We have to prove that for any $(\Sigma_1 \cap \Sigma_2)$ -sentence e' , if $E_1 \cup E_2 \models_{\Sigma_1 + \Sigma_2} e'$ then $(E_1 \cup E_2) \cap \text{Sen}(\Sigma') \models_{\Sigma'} e'$. By compactness, there are finite sets of sentences $E_1^0 \subseteq E_1$ and $E_2^0 \subseteq E_2$ such that $E_1^0 \cup E_2^0 \models_{\Sigma_1 + \Sigma_2} e'$. $(E_2^0 \Rightarrow e')$ is a Σ_2 -sentence by construction, and $E_1^0 \models_{\Sigma_1 + \Sigma_2} (E_2^0 \Rightarrow e')$.

Now let $I \subseteq \text{Sen}(\Sigma')$ be an interpolant such that $E_1^0 \models_{\Sigma_1} I \models_{\Sigma_2} (E_2^0 \Rightarrow e')$. By compactness, we may assume that I is finite.

From $E_1 \models_{\Sigma_1} I$ it follows that $I \subseteq E_1$, because E_1 is closed. Thus $I \subseteq E_1 \cap \text{Sen}(\Sigma')$. Also, $I \models_{\Sigma_2} (E_2^0 \Rightarrow e')$ implies that $E_2^0 \models_{\Sigma_2} (I \Rightarrow e')$, which implies that $(I \Rightarrow e') \subseteq E_2$, because E_2 is closed. Therefore $(I \Rightarrow e') \subseteq E_2 \cap \text{Sen}(\Sigma')$.

The desired conclusion now follows from the facts that $I \cup (I \Rightarrow e') \subseteq (E_1 \cup E_2) \cap \text{Sen}(\Sigma')$ and that $I \cup (I \Rightarrow e') \models_{\Sigma'} e'$.

Now for the converse, suppose that \mathcal{I} also has false and satisfies the distributive law. Because \mathcal{I} is compact, the Craig Interpolation Property is equivalent to its finitary version (in which all of the sets of sentences in the definition of the Craig Interpolation Property are finite).

Consider finite sets E_1 of Σ_1 -sentences and E_2 of Σ_2 -sentences such that $E_1 \models_{\Sigma_1 + \Sigma_2} E_2$. Let T_1 be the Σ_1 -theory generated by E_1 and T_2 the Σ_2 -theory generated by $(E_2 \Rightarrow \text{false}_{\Sigma_2})$. Because $\text{false}_{\Sigma'} = \text{false}_{\Sigma_1 + \Sigma_2}$ lies in $T_1 + T_2$, it follows that $\text{false}_{\Sigma'}$ belongs to $\Sigma' \square T_1 + \Sigma' \square T_2$.

There are finite sets of sentences E_1^0 in $\Sigma' \square T_1$ and E_2^0 in $\Sigma' \square T_2$ such that $E_1^0 \cup E_2^0 \models_{\Sigma'} \text{false}$. We claim that E_1^0 will serve as an interpolant. Because T_1 is generated by E_1 we have that $E_1 \models_{\Sigma_1} E_1^0$. From $E_1^0 \cup (E_2 \Rightarrow \text{false}) \models_{\Sigma'} \text{false}$, it follows that $E_1^0 \models_{\Sigma_2} E_2$. \square

It may be worth noting that in proving this result, we could assume closure under arbitrary implication, instead of compactness and closure under finite implication. But this would be less realistic, because computing systems only handle finite amounts of data.

There is an apparently stronger formulation of the middle distributive law that is actually equivalent under certain conditions:

Theorem 50 A reasonable distributive institution with Craig Interpolation satisfies the (middle) distributive law iff for every Σ_1 -theory T_1 , Σ_2 -theory T_2 , and signature Σ ,

$$(\Sigma_1 \cap \Sigma_2) \subseteq \Sigma \text{ implies } \Sigma \square (T_1 + T_2) = (\Sigma \square T_1) + (\Sigma \square T_2).$$

We will call this property the **Amsterdam distributive law**.

Proof: The middle distributive law is the special case where $\Sigma = \Sigma_1 \cap \Sigma_2$, so we only need to show that the strong form follows from the weak form.

Let Σ_1 , Σ_2 and Σ be signatures such that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma$, and let (Σ_1, E_1) and (Σ_2, E_2) be theories. Then we must prove that

$$\Sigma \square (\Sigma_1, E_1) + \Sigma \square (\Sigma_2, E_2) = \Sigma \square ((\Sigma_1, E_1) + (\Sigma_2, E_2)).$$

A simple computation shows that the signature of each side of this equality is $\Sigma \cap (\Sigma_1 + \Sigma_2)$, which we will denote Σ' . The equality can be therefore rewritten as

$$(\Sigma', (\text{Sen}(\Sigma') \cap (E_1 \cup E_2))^\bullet) = (\Sigma', \text{Sen}(\Sigma') \cap (E_1 \cup E_2)^\bullet).$$

Because the lefthand side is clearly included in the righthand side, we need only prove the opposite inclusion, that is, if $E_1 \cup E_2 \models_{\Sigma_1 + \Sigma_2} e$ for $e \in \text{Sen}(\Sigma')$, then $\text{Sen}(\Sigma') \cap (E_1 \cup E_2) \models_{\Sigma'} e$. Let $\Sigma'_1 = \Sigma_1 + (\Sigma \cap \Sigma_2)$ and let $\Sigma'_2 = \Sigma_2 + (\Sigma \cap \Sigma_1)$. Then $\Sigma'_1 \cap \Sigma'_2 = (\Sigma_1 + (\Sigma \cap \Sigma_2)) \cap (\Sigma_2 + (\Sigma \cap \Sigma_1)) = (\Sigma + \Sigma_1) \cap (\Sigma_1 + \Sigma_2) \cap (\Sigma + \Sigma_2) \cap (\Sigma_1 + \Sigma_2) = (\Sigma + (\Sigma_1 \cap \Sigma_2)) \cap (\Sigma_1 + \Sigma_2)$. Because $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma$, we have that $\Sigma'_1 \cap \Sigma'_2 = \Sigma \cap (\Sigma_1 + \Sigma_2) = \Sigma'$.

Now let E'_1 be the Σ'_1 -theory generated by E_1 , let E'_2 be the Σ'_2 -theory generated by E_2 , and suppose that $E_1 \cup E_2 \models_{\Sigma_1 + \Sigma_2} e$ with $e \in \text{Sen}(\Sigma')$. Because $E_1 \subseteq E'_1$ and $E_2 \subseteq E'_2$, we also have that $E'_1 \cup E'_2 \models_{\Sigma_1 + \Sigma_2} e$. Now we are in a position to apply the middle distributive law for (Σ'_1, E'_1) and (Σ'_2, E'_2) , which gives $\text{Sen}(\Sigma') \cap (E'_1 \cup E'_2) \models_{\Sigma'} e$.

Next we show that $\text{Sen}(\Sigma') \cap (E_1 \cup E_2) \models_{\Sigma'} E'_1 \cap \text{Sen}(\Sigma')$. Pick any Σ' -sentence e' in E'_1 . Because E'_1 is the Σ'_1 -theory generated by E_1 , we have that $E_1 \models_{\Sigma'_1} e'$. But $\Sigma'_1 = \Sigma_1 + (\Sigma \cap \Sigma_2) = \Sigma_1 + \Sigma'$, while E_1 is a set of Σ_1 -sentences, and e' is a Σ' -sentence. So by the Craig Interpolation Property, we get $I \subseteq \text{Sen}(\Sigma' \cap \Sigma_1) = \text{Sen}(\Sigma') \cap \text{Sen}(\Sigma_1)$ such that $E_1 \models_{\Sigma_1} I$ and $I \models_{\Sigma'} e'$. Because E_1 is a theory and $I \subseteq \text{Sen}(\Sigma_1)$, it follows that

$I \subseteq E_1$. Because $I \subseteq \text{Sen}(\Sigma')$ as well, $I \subseteq E_1 \cap \text{Sen}(\Sigma')$. Thus, $E_1 \cap \text{Sen}(\Sigma') \models_{\Sigma'} e'$, which means that $(E_1 \cup E_2) \cap \text{Sen}(\Sigma') \models_{\Sigma'} e'$.

Similarly, we obtain that $(E_1 \cup E_2) \cap \text{Sen}(\Sigma') \models_{\Sigma'} E'_2 \cap \text{Sen}(\Sigma')$.

Therefore $(E_1 \cup E_2) \cap \text{Sen}(\Sigma') \models_{\Sigma'} (E'_1 \cup E'_2) \cap \text{Sen}(\Sigma') \models_{\Sigma'} e$. \square

The Amsterdam distributive law appears in the work of Bergstra, Heering and Klint [4], who note that it has the form of a conditional equation; notice also that the middle distributive law is a special case.

Corollary 51 In a reasonable distributive institution that is closed under implication and has false, the followings are equivalent:

1. the Craig Interpolation Property;
2. the middle distributive law;
3. the Amsterdam distributive law.

Proof: This follows from Theorem 50 and Theorem 49. \square

The proof of Theorem 49 resembles the proof of equivalence of the distributive law and the Craig Interpolation Property given in [4] for first order logic. Indeed, we can apply the above result to obtain the distributive law for first order logic, because first order logic is compact, closed under implication and false, and satisfies the Craig Interpolation Property. However, equational logic, and also conditional equational logic, are not closed under implication and false, so that we cannot obtain the distributive law in this way for these logical systems. In fact, the distributive law does not hold for them, as shown by the following counterexample, adapted from [4]:

Example 52 Consider the unsorted signatures Σ_1 containing the constants 0 , c_1 and c_2 and the unary operation symbol $-$, and Σ_2 containing the constants c_1 and c_2 , the unary operation symbol $-$, and the ternary operation symbol h .

Let T_1 be the Σ_1 -theory generated by the equation $-0 = 0$ and let T_2 be the Σ_2 -theory generated by the equations $h(x, x, y) = y$, $h(x, -x, c_1) = h(x, -x, c_2)$ and $-(-x) = x$. The signature $\Sigma = \Sigma_1 \cap \Sigma_2$ contains only c_1 and c_2 as constants and the unary operation symbol $-$. $T_1 + T_2$ contains the equation $c_1 = c_2$ and so does $\Sigma \square (T_1 + T_2)$. Notice that $\Sigma \square T_1$ is empty and that $\Sigma \square T_2$ does not contain $c_1 = c_2$. Therefore $c_1 = c_2$ does not belong to $(\Sigma \square T_1) + (\Sigma \square T_2)$. \square

A simpler example could be given along the lines of one in [27] showing the need for explicit quantification in many sorted equational logic with possibly empty carriers. However, the example given above is stronger, because it shows that the (middle) distributive law does not hold for unsorted equational logic. We will see in Section 4.3 that this law does hold when the theory inclusions are conservative extensions, and we will argue that this covers the expected applications.

4.2 The Weak Distributive Law

This subsection shows that the weak distributive law holds for all of the institutions that are generally used for specification in Computing Science, by reducing it to a condition that is easy to check.

Definition 53 An inclusive institution \mathcal{I} is **uniform** iff for any inclusion of signatures $\Sigma' \hookrightarrow \Sigma$, any set E' of Σ' -sentences, and any Σ' -sentence e' , if $E' \models_{\Sigma} e'$ then $E' \models_{\Sigma'} e'$. \square

Intuitively, this condition says that any semantic consequence e also follows using only the symbols that actually occur in it. Also, note that we can apply this definition to a syntactic consequence relation by using Theorem 35 of Section 2.4.

Proposition 54 An inclusive institution satisfies the weak distributive law iff it is uniform.

Proof: Consider a signature Σ' which is included in both Σ_1 and Σ_2 . Let E_2 be a theory over Σ_2 . Now we calculate the two sides of the weak distributive law. First,

$$\Sigma' \square ((\Sigma_1, \emptyset^\bullet) + (\Sigma_2, E_2)) = \Sigma' \square (\Sigma_1 + \Sigma_2, E_2^\bullet) = (\Sigma', \text{Sen}(\Sigma') \cap E_2^\bullet),$$

which is a theory by Proposition 41; notice that the closure of E_2 is taken in $\Sigma_1 + \Sigma_2$. Next,

$$(\Sigma' \square (\Sigma_1, \emptyset^\bullet)) + (\Sigma' \square (\Sigma_2, E_2)) = (\Sigma', \emptyset^\bullet) + (\Sigma', E_2 \cap \text{Sen}(\Sigma')) = (\Sigma', E_2 \cap \text{Sen}(\Sigma')).$$

Clearly, we have the inclusion $E_2 \cap \text{Sen}(\Sigma') \subseteq \text{Sen}(\Sigma') \cap E_2^\bullet$.

We next show that the opposite inclusion is equivalent to uniformity. Suppose that uniformity holds for the inclusion $\Sigma_2 \hookrightarrow \Sigma_1 + \Sigma_2$. Pick a Σ_2 -sentence e' such that $E_2 \models_{\Sigma_1 + \Sigma_2} e'$. Then $E_2 \models_{\Sigma_2} e'$, which implies that $\text{Sen}(\Sigma_2) \cap E_2^\bullet \subseteq E_2$. The desired inequality follows by intersecting this inequality with $\text{Sen}(\Sigma')$.

The converse follows from the particular case where $\Sigma_2 \hookrightarrow \Sigma_1$. The weak distributive law (or more precisely, the inclusion mentioned above) now gives uniformity for $\Sigma_2 \hookrightarrow \Sigma_1$. \square

The logical systems that are widely used in Computing Science are uniform (under their usual notion of inclusion). The rest of this subsection gives some machinery for establishing the uniformity of institutions.

Definition 55 An inclusive institution **has model expansion** iff for every signature inclusion $\Sigma' \hookrightarrow \Sigma$ and for every Σ' -model M' there is a Σ -model M such that $M \upharpoonright_{\Sigma'} = M'$. \square

In the applications, this property is obtained by interpreting the symbols of Σ that are not in Σ' in an arbitrary way in the model M' . This can be done only if we do not have empty carriers in the models. This can be assured by assuming the existence of at least one constant for any sort which is the target of a new operation symbol in Σ' . Model expansion is used, for example, by Rodenburg [50] in his elegant proof of the Craig Interpolation Property for conditional equational logic.

Lemma 56 Any inclusive institution with model expansion is uniform.

Proof: Let $\Sigma' \hookrightarrow \Sigma$ be an inclusion of signatures. Let E' be a set of Σ' -sentences and let e' be a single Σ' -sentence such that $E' \models_{\Sigma'} e'$. Our aim is to prove that $E' \models_{\Sigma} e'$.

Pick a Σ' -model M' of E' . Let M be a Σ -model such that $M \upharpoonright_{\Sigma'} = M'$. By the Satisfaction Condition, $M \models_{\Sigma} E'$. It follows that $M \models_{\Sigma} e'$. Going backwards with the Satisfaction Condition, we obtain $M' \models_{\Sigma'} e'$. This implies that $E' \models_{\Sigma'} e'$. \square

Corollary 57 An inclusive institution satisfies the weak distributive law if it has model expansion. \square

Thus, the weak distributive law is closely related to model expansion and uniformity, and is not very closely related to the Craig Interpolation Property.

4.3 The Distributive Law and Conservative Extensions

We have seen that the (middle) distributive law does not hold for equational logics. This subsection shows that it does hold for semiexact reasonable institutions, including equational logics, when only conservative extensions are used.

Definition 58 A theory morphism $\phi: (\Sigma, E) \rightarrow (\Sigma', E')$ is **conservative** iff for each (Σ, E) -model M there is a (Σ', E') -model M' such that M is the ϕ -reduct of M' (i.e., $M' \upharpoonright_{\phi} = M$). Also ϕ is a **conservative extension** iff it is an extension and is conservative. \square

We claim that conservative extensions are the case of most interest for applications to hardware and/or software systems, because good design practice demands that an imported subsystem should not exhibit behaviour in its new context that differs from what its specification says it should do, i.e., the inclusion into a larger system should be conservative. Although rare in software and hardware design, non-conservative extension can easily arise in theorem proving; for example, we might form the integers modulo 2 from the integers by adding the equation $2 = 0$, or form commutative groups from groups by adding the commutativity axiom.

Theorem 59 In a semiexact reasonable institution, if $T_1 = (\Sigma_1, E_1)$ and $T_2 = (\Sigma_2, E_2)$ are theories, and if the extensions $(\Sigma_1 \cap \Sigma_2) \sqcup T_1 \hookrightarrow T_1$ and $(\Sigma_1 \cap \Sigma_2) \sqcup T_2 \hookrightarrow T_2$ are conservative, then the distributive law holds for the theories T_1 and T_2 .

Proof: Let Σ' denote $\Sigma_1 \cap \Sigma_2$. By the computations in the proof of Theorem 49, we see that the distributive law for T_1 and T_2 is equivalent to

$$(\Sigma', \text{Sen}(\Sigma') \cap (E_1 \cup E_2)^{\bullet}) \subseteq (\Sigma', ((E_1 \cup E_2) \cap \text{Sen}(\Sigma'))^{\bullet}).$$

Let e' be a Σ' -sentence such that $E_1 \cup E_2 \models_{\Sigma_1 + \Sigma_2} e'$ and pick a Σ' -model M' of $\text{Sen}(\Sigma') \cap (E_1 \cup E_2)$. Because M' is a model for $\text{Sen}(\Sigma') \cap E_1$ and because $\Sigma' \sqcup T_1 \hookrightarrow T_1$ is conservative, there is a Σ_1 -model M_1 that satisfies E_1 and whose reduct $M_1 \upharpoonright_{\Sigma'}$ is M' . Similarly, there is a Σ_2 -model M_2 that satisfies E_2 and whose reduct $M_2 \upharpoonright_{\Sigma'}$ is M' .

Because Mod preserves pushouts and $(\Sigma_1 + \Sigma_2)$ is the pushout of $\Sigma' \hookrightarrow \Sigma_1$ and $\Sigma' \hookrightarrow \Sigma_2$ (by Proposition 15), $\text{Mod}(\Sigma_1 + \Sigma_2)$ is the pullback of $\text{Mod}(\Sigma_1) \rightarrow \text{Mod}(\Sigma')$ and $\text{Mod}(\Sigma_2) \rightarrow \text{Mod}(\Sigma')$. Then there is a $(\Sigma_1 + \Sigma_2)$ -model M such that $M \upharpoonright_{\Sigma_1} = M_1$ and $M \upharpoonright_{\Sigma_2} = M_2$. By the Satisfaction Condition, $M \models_{\Sigma_1 + \Sigma_2} E_1$ and $M \models_{\Sigma_1 + \Sigma_2} E_2$. Therefore $M \models_{\Sigma_1 + \Sigma_2} e'$. Finally, because $M' = M \upharpoonright_{\Sigma}$, we conclude that $M' \models_{\Sigma} e'$. \square

4.4 Summary Concerning the Distributive Laws

Let us now summarise the situation for the distributive laws. The condition that one might really wish for, the strong distributive law, holds in none of the institutions that are generally used for specification in Computing Science. This corresponds to the intuition and experience of designers and users, that not all of the properties of a complex system that are visible through some interface are explicable without reference to the internals of the system. However, a weaker version of the distributive law, here called the middle distributive law, does hold for some institutions of interest. In our view, this is not a particularly significant property, because its hypothesis unrealistically requires the external visibility of all internal interfaces of a system. Note that it is not in general equivalent to the Craig Interpolation Property; for example, equational logic satisfies Craig interpolation but not the middle distributive law. However, these two conditions are equivalent for institutions with sufficiently rich resources for combining sentences, and they are equivalent to a strong form of the middle distributive law. Also, at least for first order logic, the middle distributive law supports a normal form for module expressions, in which information hiding can be reduced to a single operation which is performed last [4]. Subsection 4.3 showed that the middle distributive law holds for equational style logics when the extensions are conservative, and argued that this is the case of greatest interest in applications. Finally, let us recall that the weak distributive law is so weak that it holds in all of the institutions that are widely used in Computing Science.

4.5 The Algebra of Model Classes

We now explore some relationships between denotations of modules as sets of sentences, and as classes of models, assuming a reasonable institution \mathcal{I} . Recall that if P is a presentation, then $\llbracket P \rrbracket$ denotes its class of models.

From its very beginning in the work of Parnas [47] and others, information hiding has been a syntactic notion: it refers to scoping conventions that prevent the use of certain parts of a module that are regarded as internal; the purpose is to make it easier to reuse and maintain software. Similarly, a proof in mathematics often contains many details that are hidden in the statement of the result proved; if these details were brought to the surface, it would make reuse of the result much more difficult. The semantic counterpart of information hiding is the reduct operation; note that in general we cannot expect to see all the details of all the models of large systems, because we can only work with their descriptions. Nevertheless, it is interesting to know how the two kinds of information hiding relate. We also consider a semantic version of sum.

Definition 60 Let C_1 be a collection of Σ_1 -models and C_2 be a collection of Σ_2 -models. Then $C_1 + C_2$ is defined to be the collection of those $(\Sigma_1 + \Sigma_2)$ -models M such that $M \upharpoonright_{\Sigma_1} \in C_1$ and $M \upharpoonright_{\Sigma_2} \in C_2$.

Let Σ be a signature and let C' be a class of Σ' -models. Then $\Sigma \square C'$ is the collection of $(\Sigma \cap \Sigma')$ -models $\{M \upharpoonright_{\Sigma \cap \Sigma'} \mid M \in C'\}$. \square

We first show that sum commutes with denotation:

Fact 61 $\llbracket T_1 + T_2 \rrbracket = \llbracket T_1 \rrbracket + \llbracket T_2 \rrbracket$ for any theories T_1 and T_2 .

Proof: this is direct from the definitions and Corollary 27. \square

However, the non-commutativity of denotation with some other operations highlights the difference between the algebra of theories and the algebra of model classes.

Proposition 62 Let T be a theory and Σ a signature. Then $\Sigma \square [T] \subseteq \llbracket \Sigma \square T \rrbracket$, with equality iff the extension $\Sigma \square T \hookrightarrow T$ is conservative.

Proof: Let $T = (\Sigma', E)$ and let Σ be any signature. Because $\Sigma \square T = (\Sigma \cap \Sigma') \square T$ (by Definition 40), and $\Sigma \square [T] = (\Sigma \cap \Sigma') \square [T]$ (by Definition 60), we can replace Σ by $\Sigma \cap \Sigma'$. And because $\Sigma \cap \Sigma' \hookrightarrow \Sigma'$, we may assume that $\Sigma \hookrightarrow \Sigma'$. Now let $M \in \Sigma \square [T]$, which means that $M = M' \upharpoonright_{\Sigma}$ for some $M' \models_{\Sigma'} E$. Then $M' \models_{\Sigma'} E \cap \text{Sen}(\Sigma)$ and so by the Satisfaction Condition, $M' \upharpoonright_{\Sigma} \models_{\Sigma} E \cap \text{Sen}(\Sigma)$, i.e., $M \in \llbracket \Sigma \square T \rrbracket$.

The equality $\Sigma \square [T] = \llbracket \Sigma \square T \rrbracket$ holds iff for any model M of $E \cap \text{Sen}(\Sigma)$ there is a model M' of E such that $M = M' \upharpoonright_{\Sigma}$. \square

We now give an example showing that this inclusion can be strict.

Example 63 Let Σ' be the unsorted signature with two constants c and 0 and an unary operation symbol $-$. Let T be the Σ' -theory generated by the equation $0 = -0$. Let Σ be the signature containing only the constant c and the unary operation symbol $-$. Then $\Sigma \square T$ is empty, and so its denotation $\llbracket \Sigma \square T \rrbracket$ is $\text{Mod}(\Sigma)$. Now consider the Σ -model M having $\{1, 2\}$ as its underlying set and interpreting c as 1, and $-1 = 2$ and $-2 = 1$. Then there is no model of T such that its Σ -reduct is M , because any such model must have at least one fixpoint for $-$. Thus $\Sigma \square [T]$ is strictly included in $\llbracket \Sigma \square T \rrbracket$. \square

Corollary 64 Let T be a theory and Σ be a signature. Then $\Sigma \square T \subseteq (\Sigma \square [T])^*$, with equality iff the extension $\Sigma \square T \hookrightarrow T$ is conservative. \square

This result says that information hiding on denotations is more accurate than on theories. Because of this, we should not be surprised to see that the middle distributive law holds for model classes. But we might still be surprised at the length of the proof.

Proposition 65 A semiexact reasonable distributive institution satisfies the Amsterdam distributive law for denotations of theories; i.e., given theories $T_1 = (\Sigma_1, E_1)$ and $T_2 = (\Sigma_2, E_2)$, and given $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma$, then

$$\Sigma \square (\llbracket T_1 \rrbracket + \llbracket T_2 \rrbracket) = \Sigma \square \llbracket T_1 \rrbracket + \Sigma \square \llbracket T_2 \rrbracket.$$

Proof: Let us first compute the two sides of the equation:

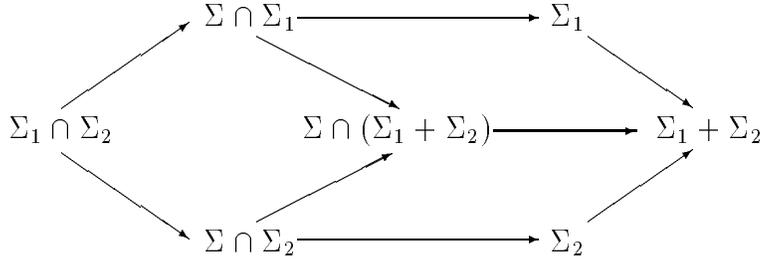
$$\begin{aligned} & \Sigma \square (\llbracket T_1 \rrbracket + \llbracket T_2 \rrbracket) \\ &= \Sigma \square \{M \in \text{Mod}(\Sigma_1 + \Sigma_2) \mid M \upharpoonright_{\Sigma_1} \in \text{Mod}(T_1) \text{ and } M \upharpoonright_{\Sigma_2} \in \text{Mod}(T_2)\} \\ &= \{M \upharpoonright_{\Sigma \cap (\Sigma_1 + \Sigma_2)} \mid M \in \text{Mod}(\Sigma_1 + \Sigma_2) \text{ and } M \upharpoonright_{\Sigma_1} \in \text{Mod}(T_1) \text{ and } M \upharpoonright_{\Sigma_2} \in \text{Mod}(T_2)\}; \end{aligned}$$

and

$$\begin{aligned} & \Sigma \square \llbracket T_1 \rrbracket + \Sigma \square \llbracket T_2 \rrbracket \\ &= \{M_1 \upharpoonright_{\Sigma \cap \Sigma_1} \mid M_1 \in \text{Mod}(T_1)\} + \{M_2 \upharpoonright_{\Sigma \cap \Sigma_2} \mid M_2 \in \text{Mod}(T_2)\} \\ &= \{N \in \text{Mod}(\Sigma \cap (\Sigma_1 + \Sigma_2)) \mid N \upharpoonright_{\Sigma \cap \Sigma_1} \text{ has an expansion } M_1 \in \text{Mod}(T_1) \text{ and } \\ & \quad N \upharpoonright_{\Sigma \cap \Sigma_2} \text{ has an expansion } M_2 \in \text{Mod}(T_2)\}. \end{aligned}$$

Now $\Sigma \square ([[T_1]] + [[T_2]]) \subseteq \Sigma \square [[T_1]] + \Sigma \square [[T_2]]$ because for any model $N \in \text{Mod}(\Sigma \cap (\Sigma_1 + \Sigma_2))$ such that $N = M \upharpoonright_{\Sigma \cap (\Sigma_1 + \Sigma_2)}$ for some $M \in \text{Mod}(\Sigma_1 + \Sigma_2)$ with $M \upharpoonright_{\Sigma_1} \in \text{Mod}(T_1)$ and $M \upharpoonright_{\Sigma_2} \in \text{Mod}(T_2)$, we have that $M \upharpoonright_{\Sigma_i}$ is an expansion of $N \upharpoonright_{\Sigma \cap \Sigma_i}$ for $i = 1, 2$.

The argument for the opposite inclusion may be aided by the diagram below, in which all arrows are inclusions. First observe that $\Sigma_1 \cap \Sigma_2 = (\Sigma \cap \Sigma_1) \cap (\Sigma \cap \Sigma_2)$. Because the institution is distributive, we can apply Proposition 15 to conclude that $\Sigma \cap \Sigma_1 \hookrightarrow \Sigma \cap (\Sigma_1 + \Sigma_2)$ and $\Sigma \cap \Sigma_2 \hookrightarrow \Sigma \cap (\Sigma_1 + \Sigma_2)$ are the pushout of $\Sigma_1 \cap \Sigma_2 \hookrightarrow \Sigma \cap \Sigma_1$ and $\Sigma_1 \cap \Sigma_2 \hookrightarrow \Sigma \cap \Sigma_2$. Now consider a model $N \in \text{Mod}(\Sigma \cap (\Sigma_1 + \Sigma_2))$ such that $N \upharpoonright_{\Sigma \cap \Sigma_1}$ has an extension $M_1 \in \text{Mod}(T_1)$ and $N \upharpoonright_{\Sigma \cap \Sigma_2}$ has an extension $M_2 \in \text{Mod}(T_2)$. Then $M_1 \upharpoonright_{\Sigma_1 \cap \Sigma_2} = M_2 \upharpoonright_{\Sigma_1 \cap \Sigma_2} = N \upharpoonright_{\Sigma_1 \cap \Sigma_2}$. Because the institution is semiexact, there is a model $M \in \text{Mod}(\Sigma_1 + \Sigma_2)$ such that $M \upharpoonright_{\Sigma_1} = M_1$ and $M \upharpoonright_{\Sigma_2} = M_2$. All that remains to show is that $M \upharpoonright_{\Sigma \cap (\Sigma_1 + \Sigma_2)} = N$. But $(M \upharpoonright_{\Sigma \cap (\Sigma_1 + \Sigma_2)}) \upharpoonright_{\Sigma \cap \Sigma_i} = (M \upharpoonright_{\Sigma_i}) \upharpoonright_{\Sigma \cap \Sigma_i} = M_i \upharpoonright_{\Sigma \cap \Sigma_i} = N \upharpoonright_{\Sigma \cap \Sigma_i}$ for $i = 1, 2$, which shows that $M \upharpoonright_{\Sigma \cap (\Sigma_1 + \Sigma_2)} = N$. \square



Corollary 66 A reasonable semiexact institution satisfies the middle distributive law for denotations of theories; i.e., for any theories $T_1 = (\Sigma_1, E_1)$ and $T_2 = (\Sigma_2, E_2)$, if $\Sigma = \Sigma_1 \cap \Sigma_2$, then

$$\Sigma \square ([[T_1]] + [[T_2]]) = \Sigma \square [[T_1]] + \Sigma \square [[T_2]].$$

Proof: This follows from the previous result with $\Sigma = \Sigma_1 \cap \Sigma_2$, noting that in this case the proof does not need distributivity. \square

5 Pushouts of Extensions

The concept of conservative extension can be defined either semantically or syntactically; the syntactic condition is necessary, but not sufficient for the semantic condition. Byers and Pitt [8] give a nice discussion of the problem with some counterexamples, and Veloso and Veloso [65] discuss counterexamples from first order logic in more detail. The fact that models are the objects of primary interest in Computing Science applications suggests that the model theoretic definition of conservative extension should be given priority, and the fact that the syntactic formulation does not exactly characterise what happens to the models seems to us evidence against taking a purely deductive approach to the foundations of software engineering, as advocated by Maibaum, Turski, Sadler and others [40, 64]. However, this does not mean that π -institutions should be abandoned; for example, they may be appropriate for deductive databases (as was suggested in [16]), where it seems reasonable⁶ to take theories as models, as in the construction of Theorem 35.

⁶This is based on the view that a database is a self-contained hypothetical world, which may or may not correspond to “reality.” Actually, large databases are rarely completely accurate, because of typing errors, social and political issues, delayed updates, and a wide variety of other causes.

This section considers pushouts of three different variants of the notion of conservative extension. The first is the notion in Definition 58 of the previous section. Then we consider a stronger form of conservatism called persistency, that is relevant to parameterised modules. Finally, we consider extensions that conserve initial models of the theories involved; this is relevant to the specification of data types. Similar results have been previously obtained for the equational case by Ehrig, Kreowski, Thatcher, Wagner and Wright [15], by Goguen and Meseguer [26], and by others; early work extending these concepts to institutions was done by Goguen and Burstall in [23].

5.1 Conservative Extensions

This section shows that pushouts of conservative extensions are conservative under some weak assumptions.

Proposition 67 Given a semiexact institution with pushouts of signatures, let (ϕ', ψ') be a pushout of the theory morphisms $\phi: P \rightarrow T$ and $\psi: P \rightarrow P'$. Then $\phi': P' \rightarrow T'$ is conservative if $\phi: P \rightarrow T$ is conservative.

Proof: Suppose that $\phi: P \rightarrow T$ is conservative, and pick an arbitrary model N' of P' . Then $N' \upharpoonright_{\psi}$ is a model of P by the Satisfaction Condition. Because ϕ is conservative, there is a model M of T such that $M \upharpoonright_{\phi} = N' \upharpoonright_{\psi}$. By Proposition 30, there is a model M' of T' such that $M' \upharpoonright_{\psi'} = M$ and $M' \upharpoonright_{\phi'} = N'$. Because N' was arbitrary, we conclude that any P' -model has an ϕ' -expansion to a T' -model.

$$\begin{array}{ccc}
 P & \xrightarrow{\phi} & T \\
 \psi \downarrow & & \downarrow \psi' \\
 P' & \xrightarrow{\phi'} & T'
 \end{array}$$

□

Corollary 68 In any semiexact strongly inclusive institution where pushouts preserve signature inclusions, pushouts also preserve conservative theory extensions.

Proof: We can lift the preservation of inclusions by pushouts from signatures to theories by Fact 32, and then apply Proposition 67. □

The following notion is often found in the literature. We will show that it is a necessary but insufficient condition for conservation:

Definition 69 A theory extension $(\Sigma, E) \hookrightarrow (\Sigma', E')$ is **syntactically conservative** iff $E = E' \cap \text{Sen}(\Sigma)$. □

Proposition 70 If a theory extension $(\Sigma, E) \hookrightarrow (\Sigma', E')$ is conservative, then it is syntactically conservative.

Proof: It suffices to show that $E' \cap \text{Sen}(\Sigma) \subseteq E$, because $E \subseteq E'$ already. So let $e' \in E' \cap \text{Sen}(\Sigma)$; then it suffices to show $e' \in E$.

Let $M \models_{\Sigma} E$. Then there is an M' such that $M \models_{\Sigma'} E'$ and $M' \upharpoonright_{\Sigma} = M$. Therefore $M' \models_{\Sigma'} e'$. Thus, $M' \upharpoonright_{\Sigma} \models_{\Sigma} e'$, i.e., $M \models_{\Sigma} e'$. Therefore $e' \in E^{\bullet} = E$. □

Maibaum and Sadler [40] gave an example showing that syntactic conservatism is not preserved under pushouts in the equational institution. From this, they concluded that equational logic is defective. However, we conclude instead that syntactic conservatism is not a sufficient condition for true conservatism. For if it were, then Corollary 68 and the above proposition would imply that pushouts preserve syntactic conservatism in any semiexact strongly inclusive institution, including equational logic, which their example shows is false. From this we conclude that the syntactic definition of conservatism is defective. We have already given a similar example in this paper: because an extension $(\Sigma, E) \rightarrow (\Sigma', E')$ is syntactically conservative iff $\Sigma \sqcap (\Sigma', E') = (\Sigma, E)$, Example 63 provides an extension $\Sigma \sqcap (\Sigma', E') \rightarrow (\Sigma', E')$ that is syntactically but not semantically conservative.

5.2 Persistent Extensions

Persistence is a stronger notion than conservative extension, and is important for the semantics of parameterised data types (e.g., see [23]).

Definition 71 A theory morphism $\phi: P \rightarrow T$ is **persistent** iff its associated reduct functor $-\upharpoonright_\phi: Mod(T) \rightarrow Mod(P)$ has a left adjoint such that each component of the unit of the adjunction is an equality. \square

Fact 72 A persistent theory morphism is conservative. \square

The following result is related to the semantics of applying a generic module to an actual parameter module using a “view,” as proposed in Clear and implemented in OBJ3:

Proposition 73 Given a semiexact institution with pushouts of signatures, let (ϕ', ψ') be the pushout of theory morphisms $\phi: P \rightarrow T$ and $\psi: P \rightarrow P'$. Then:

1. If the functor $-\upharpoonright_\phi: Mod(T) \rightarrow Mod(P)$ has a left inverse $\phi^{\mathfrak{s}}: Mod(P) \rightarrow Mod(T)$, then there is a left inverse $\phi'^{\mathfrak{s}}$ of $-\upharpoonright_{\phi'}$ such that the following diagram commutes:

$$\begin{array}{ccc}
 Mod(P) & \xrightarrow{\phi^{\mathfrak{s}}} & Mod(T) \\
 \uparrow -\upharpoonright_\psi & & \uparrow -\upharpoonright_{\psi'} \\
 Mod(P') & \xrightarrow{\phi'^{\mathfrak{s}}} & Mod(T')
 \end{array}$$

2. ϕ' is persistent if ϕ is persistent.

Proof: To show the first assertion, pick an arbitrary model N' of P' . Then $N = N' \upharpoonright_\psi$ is a model of P by the Satisfaction Condition. Let M be $\phi^{\mathfrak{s}}(N)$. Then $M \upharpoonright_\phi = N' \upharpoonright_\psi = N$. By Proposition 30, there is a model M' of T' such that $M' \upharpoonright_{\psi'} = M$ and $M' \upharpoonright_{\phi'} = N'$. The mapping $N' \mapsto M'$ defines the functor $\phi'^{\mathfrak{s}}$ on objects, and its definition on arrows is similar. Next, $\phi'^{\mathfrak{s}}$ preserves identities because $1_{M'} \upharpoonright_{\psi'} = \phi'^{\mathfrak{s}}(1_{N'}) \upharpoonright_{\psi'}$ and $1_{M'} \upharpoonright_{\phi'} = \phi'^{\mathfrak{s}}(1_{N'}) \upharpoonright_{\phi'}$ for any P' -model N' . By Proposition 30, $1_{M'} = \phi'^{\mathfrak{s}}(1_{N'})$. The same argument gives the preservation of composition by $\phi'^{\mathfrak{s}}$.

For the second assertion, we will show that $\phi^{\mathfrak{s}}$ is left-adjoint to $_ \uparrow_{\phi'}$ if $\phi^{\mathfrak{s}}$ is left-adjoint to $_ \uparrow_{\phi}$, that is (using the above notations), M' is a free T' -model over N' if M is a free T -model over N . Pick an arbitrary T' -model M'_1 and an arbitrary model morphism $h : N' \rightarrow M'_1 \uparrow_{\phi'}$. We have to prove that there is a unique model morphism $h^{\#} : M' \rightarrow M'_1$ such that $h^{\#} \uparrow_{\phi'} = h$. Notice that by Proposition 30, any $h^{\#} : M' \rightarrow M'_1$ is uniquely determined by its reducts $h = h^{\#} \uparrow_{\phi'} : N' \rightarrow M'_1 \uparrow_{\phi'}$ and $f = h^{\#} \uparrow_{\psi'} : M \rightarrow M'_1 \uparrow_{\psi'}$ and by the condition $h \uparrow_{\psi} = f \uparrow_{\phi}$.

Now let f be the unique model morphism $M \rightarrow M'_1 \uparrow_{\psi'}$ such that $h \uparrow_{\psi} = f \uparrow_{\phi}$ (since M is free over N). Then the morphism $h^{\#} : M' \rightarrow M'_1$ determined by (f, h) is the desired extension of h to a model morphism $M' \rightarrow M'_1$. \square

Corollary 74 If pushouts preserve signature inclusions in a semiexact strongly inclusive institution, then pushouts also preserve persistent theory extensions.

Proof: We can lift the preservation of inclusions by pushouts from signatures to theories by Fact 32, and then apply Proposition 73. \square

5.3 Conservative for Initiality

A rather different notion of conservative extension is appropriate when considering data types that are defined by initial model semantics.

Definition 75 A theory morphism $\phi : P \rightarrow T$ is **conservative for initiality** iff the theories P and T admit initial models, 0_P and 0_T , respectively, such that $0_T \uparrow_{\phi} = 0_{P'}$. \square

Fact 76 If both theories P and T of a persistent morphism $\phi : P \rightarrow T$ admit initial models, then ϕ is conservative for initiality.

Proof: This follows from the preservation of initial objects (which are colimits) by left adjoint functors. \square

Proposition 77 Given a semiexact institution with pushouts of signatures, let (ϕ', ψ') be the pushout of theory morphisms $\phi : P \rightarrow T$ and $\psi : P \rightarrow P'$. If both ϕ and ψ are conservative for initiality, then the pushout morphisms ϕ' and ψ' are also conservative for initiality.

Proof: Because $0_T \uparrow_{\phi} = 0_{P'} \uparrow_{\psi}$ by Proposition 30, 0_T and $0_{P'}$ determine a T' -model $0_{T'}$ such that $0_{T'} \uparrow_{\psi'} = 0_T$ and $0_{T'} \uparrow_{\phi'} = 0_{P'}$. Then initiality of $0_{T'}$ follows from the initiality of 0_T and $0_{P'}$ and the fact that any T' -morphism $0_{T'} \rightarrow M'$ is uniquely determined by its reducts $0_T \rightarrow M' \uparrow_{\psi'}$ and $0_{P'} \rightarrow M' \uparrow_{\phi'}$. \square

6 Conclusions

In order to compare the properties of different logical systems, it is necessary to employ some formalisation of the notion of logical system. Otherwise, it is difficult to be sure that general properties have been formulated correctly, and it is impossible to rigorously compare relationships among different general properties. We have used institutions to formalise logical systems. An alternative formalisation would have been π -institutions. But these leave out the vital connection with models; also, this paper shows that π -institutions can be seen as a subcategory of ordinary institutions.

We have been particularly interested in the comparison between equational logic and classical first order logic. Some authors have argued that certain properties of equational logic render it unsuitable for use in specification. These include the following:

1. the failure of pushouts to preserve conservative extensions [40];
2. failure of the (middle) distributive law for information hiding over sum [4]; and
3. failure of the Strong Craig Interpolation Property [40, 41].

The first point rests on using a *syntactic* formulation of conservative extension, rather than the *semantic* formulation that we have argued correctly describes the behaviour of the models that are the real result of the design process. Indeed, the fact that pushouts do not necessarily preserve syntactically conservative extensions, even for equational logic, seems to us further confirmation of the inappropriateness of the syntactic formulation.

As argued in Section 4.3, non-conservative importation has little practical importance for software and/or hardware design, because good design practice demands that the imported material should behave as advertised in its new context. Therefore failure to satisfy the middle distributive law for all extensions should not be held against an institution.

The third point rests on a result relating the Strong Craig Interpolation Property to the preservation of conservative extensions under pushouts. However, we have argued above that the syntactic formulation of conservative extension used in this result is inappropriate; therefore Strong Craig Interpolation is not relevant.

Many of those attacking equational logic have preferred first order logic. But in fact, there are many desirable properties satisfied by equational logic that are not satisfied by classical first order logic, including the following:

1. the existence of **initial models** for all theories, and more pointedly, a close connection with the most natural models for presentations, which are the **computable algebras** or abstract data types, and more generally, with the semi-computable models.
2. **left adjoints** for all forgetful functors induced by theory morphisms (i.e., liberality), supporting **free extensions** of models; and
3. **algorithms** like term rewriting, Knuth-Bendix, and narrowing, which give equational logic a strong computational aspect, and make it especially suitable for mechanisation.

Some additional points are that equational logic can be used conveniently for the specification and verification of imperative programs [25], as well as object oriented programs [20]. Moreover, equational deduction is significantly simpler than deduction for full first order logic, and has significant advantages for computation. Thus, the case for using equational logic in Computing Science wherever it can be used seems quite strong. Of course, equational logic cannot be used for everything, but it can be used (for example) for many problems in hardware and software specification⁷.

Equational logic also seems promising as a metalanguage for describing other logics, and hence for theorem proving over arbitrary logical systems. This approach is taken in

⁷It should be noted that there are many variants of equational logic. Although the unsorted variant is the most traditional in mathematics, it is perhaps the least suitable for Computing Science applications; for example, order sorted algebra [30] is much better, because of its capabilities to handle subsorts, errors, polymorphism and overloading.

the 2OBJ system [31], a metalogical⁸ theorem prover that supports deduction in any logical system, by implementing its abstract data type of proofs in equational logic. 2OBJ builds on facilities of the OBJ3 system [?, 33]. Both 2OBJ and OBJ3 have module systems based on the approach described in this paper. Foundations for the 2OBJ approach to theorem proving require formalising the notions of deduction in a logical system, and of encoding one logical system into another; these are provided by the notion of a ruled charter (which gives rise to an institution) and a ruled charter morphism [31], which are elaborations of ideas from [22].

Given our position that the consideration of logic in Computing Science should involve a delicate balance between syntax and semantics, it is interesting to compare our intuitive discussions of information hiding and conservative extensions. In the first case, we argued that information hiding as used in practice is by its nature syntactic, since it refers to certain bits of text that are not exported. Hence, we interpret the fact that some laws are satisfied by \square on model classes, but not on theories as explaining why programs with hidden parts work correctly in practice, even though proving this may require use of the hidden material; that is, verification may require “unhiding.”

By contrast, in the case of conservative extensions, we argued that the semantic notion should have priority, because we are primarily interested in the correct operation of models in some new context. We then noted that the syntactic criterion is necessary but not sufficient to ensure this.

To try to sum up the whole paper now, we hope to have shown that institutions can be used to define a variety of properties of logical systems that support modularisation, and to clarify the relationships among them. In particular, we have shown that certain kinds of conservative extension are preserved by pushouts, under certain conditions. These results cover extensions that are conservative for initiality, and that are persistent, as well as ordinary conservative extensions. Extensions that are conservative for initiality are important for importing (abstract) data types, and our result for this case requires that both extensions be conservative for initiality. Persistent extensions are important for applications of parameterised modules, while ordinary conservative extensions are important for importing and reusing modules. We have also considered various forms of Craig interpolation, and various algebraic laws for operations on theories; in particular, we have argued that the distributive laws for information hiding over the sum of theories are most significant for applications to hardware and/or software design when the theory extensions involved are conservative, and have given a general theorem that covers this case. Many of our results involve assumptions of (semi)exactness and reasonability, which appear to hold for most institutions of interest in Computing Science. Finally, we hope to have furthered understanding of the sometimes subtle relationships between syntax and semantics in formal methods for software and/or hardware engineering.

References

- [1] Egidio Astesiano and Maura Cerioli. Commuting between institutions by simulation. Technical Report 2, University of Genova, 1990.

⁸Following [10], this means that the system supports metalevel reasoning, that is, reasoning about proofs, as well as object level reasoning.

- [2] Jon Barwise. Axioms for abstract model theory. *Annals of Mathematical Logic*, 7:221–265, 1974.
- [3] Jon Barwise and Solomon Feferman. *Model-Theoretic Logics*. Springer, 1985.
- [4] Jan Bergstra, Jan Heering, and Paul Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.
- [5] Jan Bergstra and John Tucker. Characterization of computable data types by means of a finite equational specification method. In J.W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming, Seventh Colloquium*, pages 76–90. Springer, 1980. Lecture Notes in Computer Science, Volume 81.
- [6] 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.
- [7] Rod Burstall and Joseph Goguen. The semantics of Clear, a specification language. In Dines Bjorner, editor, *Proceedings, 1979 Copenhagen Winter School on Abstract Software Specification*, pages 292–332. Springer, 1980. Lecture Notes in Computer Science, Volume 86; based on unpublished notes handed out at the Symposium on Algebra and Applications, Stefan Banach Center, Warsaw, Poland, 1978.
- [8] Paddy Byers and David Pitt. Conservative extensions: A cautionary note. *Bulletin of the European Association for Theoretical Computer Science*, 41:196–201, June 1990.
- [9] Paul M. Cohn. *Universal Algebra*. Harper and Row, 1965. Revised edition 1980.
- [10] Robert Constable and David Basin. Meta-logical frameworks. In *Proceedings, 2nd Edinburgh Workshop on Logical Frameworks*. Cambridge, to appear 1993.
- [11] Thierry Coquand and Gerard Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988.
- [12] Nicolas de Bruijn. A survey of the project AUTOMATH. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic, 1980.
- [13] Harmut Ehrig, Michael Baldamus, Felix Cornelius, and Fernando Orejas. Abstract module specifications in the framework of specification logics and applications to behavioural module specifications. Technical report, Technical University of Berlin, 1991.
- [14] Harmut Ehrig, Michael Baldamus, and Fernando Orejas. New concepts for amalgamation and extension in the framework of specification logics. Technical Report 91/05, Technical University Berlin, 1991.
- [15] Hartmut Ehrig, Hans-Jörg Kreowski, James Thatcher, Eric Wagner, and Jesse Wright. Parameter passing in algebraic specification languages. *Theoretical Computer Science*, 28:45–81, 1984. Earlier version in *Workshop on Program Specification (Aarhus)* pages 322–369, Springer, Lecture Notes in Computer Science, Volume 134, 1981.

- [16] José Fiadeiro and Amílcar Sernadas. Structuring theories on consequence. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification*, pages 44–72. Springer, 1988. Lecture Notes in Computer Science, Volume 332.
- [17] Joseph Goguen. Categorical foundations for general systems theory. In F. Pichler and R. Trappl, editors, *Advances in Cybernetics and Systems Research*, pages 121–130. Transcripta Books, 1973.
- [18] Joseph Goguen. Reusing and interconnecting software components. *Computer*, 19(2):16–28, February 1986. Reprinted in *Tutorial: Software Reusability*, Peter Freeman, editor, IEEE Computer Society, 1987, pages 251–263, and in *Domain Analysis and Software Systems Modelling*, Rubén Prieto-Díaz and Guillermo Arango, editors, IEEE Computer Society, 1991, pages 125–137.
- [19] Joseph Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991. Also, Programming Research Group Technical Monograph PRG-72, Oxford University, March 1989.
- [20] Joseph Goguen. Types as theories. In George Michael Reed, Andrew William Roscoe, and Ralph F. Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford, 1991. Proceedings of a Conference held at Oxford, June 1989.
- [21] Joseph Goguen and Rod Burstall. CAT, a system for the structured elaboration of correct programs from structured specifications. Technical Report Report CSL-118, SRI Computer Science Lab, October 1980.
- [22] Joseph Goguen and Rod Burstall. A study in the foundations of programming methodology: Specifications, institutions, charters and parchments. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Proceedings, Conference on Category Theory and Computer Programming*, pages 313–333. Springer, 1986. Lecture Notes in Computer Science, Volume 240; also, Report CSLI-86-54, Center for the Study of Language and Information, Stanford University, June 1986.
- [23] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992. Draft appears as Report ECS-LFCS-90-106, Computer Science Department, University of Edinburgh, January 1990.
- [24] Joseph Goguen and Susanna Ginali. A categorical approach to general systems theory. In George Klir, editor, *Applied General Systems Research*, pages 257–270. Plenum, 1978.
- [25] Joseph Goguen and Grant Malcolm. *Algebraic Semantics for Imperative Languages*. Draft, Programming Research Group, Oxford University, 1992.
- [26] Joseph Goguen and José Meseguer. Universal realization, persistent interconnection and implementation of abstract modules. In M. Nielsen and E.M. Schmidt, editors, *Proceedings, 9th International Conference on Automata, Languages and Programming*, pages 265–281. Springer, 1982. Lecture Notes in Computer Science, Volume 140.

- [27] Joseph Goguen and José Meseguer. Completeness of many-sorted equational logic. *Houston Journal of Mathematics*, 11(3):307–334, 1985. Preliminary versions have appeared in: *SIGPLAN Notices*, July 1981, Volume 16, Number 7, pages 24–37; SRI Computer Science Lab, Report CSL-135, May 1982.
- [28] Joseph Goguen and José Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In Douglas DeGroot and Gary Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*, pages 295–363. Prentice-Hall, 1986. An earlier version appears in *Journal of Logic Programming*, Volume 1, Number 2, pages 179–210, September 1984.
- [29] Joseph Goguen and José Meseguer. Unifying functional, object-oriented and relational programming, with logical semantics. In Bruce Shriver and Peter Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT, 1987. Preliminary version in *SIGPLAN Notices*, Volume 21, Number 10, pages 153–162, October 1986.
- [30] Joseph Goguen and José Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992. Also, Programming Research Group Technical Monograph PRG–80, Oxford University, December 1989.
- [31] Joseph Goguen, Andrew Stevens, Keith Hobley, and Hendrik Hilberdink. 2OBJ, a metalogical framework based on equational logic. *Philosophical Transactions of the Royal Society, Series A*, 339:69–86, 1992. Also in *Mechanized Reasoning and Hardware Design*, edited by C.A.R. Hoare and M.J.C. Gordon, Prentice-Hall, 1992, pages 69–86.
- [32] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. Technical Report RC 6487, IBM T.J. Watson Research Center, October 1976. In *Current Trends in Programming Methodology, IV*, Raymond Yeh, editor, Prentice-Hall, 1978, pages 80–149.
- [33] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen, editor, *Algebraic Specification with OBJ: An Introduction with Case Studies*. Cambridge, to appear 1995. Also to appear as Technical Report from SRI International.
- [34] Joseph Goguen and David Wolfram. On types and FOOPS. In Robert Meersman, William Kent, and Samit Khosla, editors, *Object Oriented Databases: Analysis, Design and Construction*, pages 1–22. North Holland, 1991. Proceedings, IFIP TC2 Conference, Windermere, UK, 2–6 July 1990.
- [35] Robert Goldblatt. *Topoi, the Categorical Analysis of Logic*. North-Holland, 1979.
- [36] Robert Harper, David MacQueen, and Robin Milner. Standard ML. Technical Report ECS-LFCS-86-2, Department of Computer Science, University of Edinburgh, 1986.
- [37] Robert Harper, Donald Sannella, and Andrzej Tarlecki. Logic representation in LF. In David Pitt, David Rydeheard, Peter Dybjer, Andrew Pitts, and Axel Poigné, editors,

- Proceedings, Conference on Category Theory and Computer Science*, pages 250–272. Springer, 1989. Lecture Notes in Computer Science, Volume 389.
- [38] Horst Herrlich and George Strecker. *Category Theory*. Allyn and Bacon, 1973.
- [39] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.
- [40] Thomas Maibaum and Martin Sadler. Axiomatising specification theory. In Hans-Jörg Kreowski, editor, *Recent Trends in Data Type Specification*, pages 171–177. Springer, 1985. Informatik-Fachberichte 116.
- [41] Thomas Maibaum and José Fiadeiro with Martin Sadler. Stepwise program development in Π -institutions. Technical report, Imperial College, 1990.
- [42] Mila Majster. Limits of the algebraic specification of abstract data types. *SIGPLAN Notices*, pages 37–42, October 1977.
- [43] José Meseguer. General logics. In H.-D. Ebbinghaus et al., editors, *Proceedings, Logic Colloquium, 1987*, pages 275–329. North-Holland, 1989.
- [44] José Meseguer and Joseph Goguen. Initiality, induction and computability. In Maurice Nivat and John Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge, 1985.
- [45] Peter Mosses. Unified algebras and institutions. In *Proceedings, Fourth Annual Conference on Logic in Computer Science*, pages 304–312. IEEE, 1989.
- [46] Mogens Nielsen and Udo Platet. Polymorphism in an institutional framework, 1986. Technical University of Denmark.
- [47] David Parnas. Information distribution aspects of design methodology. *Information Processing '72*, 71:339–344, 1972. Proceedings of 1972 IFIP Congress.
- [48] David Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the Association for Computing Machinery*, 15:1053–1058, 1972.
- [49] David Parnas. A technique for software module specification. *Communications of the Association for Computing Machinery*, 15:330–336, 1972.
- [50] Pieter-Hendrik Rodenburg. Interpolation in conditional equational logic, 1989. Preprint from Programming Research Group at the University of Amsterdam.
- [51] Pieter-Hendrik Rodenburg. A simple algebraic proof of the equational interpolation theorem. *Algebra Universalis*, 28:48–51, 1991.
- [52] Antonino Salibra and Giuseppe Scollo. A soft stairway to institutions, 1991. University of Pisa.
- [53] Donald Sannella. *Semantics, Implementation and Pragmatics of Clear, a Program Specification Language*. PhD thesis, University of Edinburgh, Computer Science Department, 1982. Report CST-17-82.

- [54] Donald Sannella and Andrzej Tarlecki. Extended ML: an institution independent framework for formal program development. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Proceedings, Summer Workshop on Category Theory and Computer Programming*, pages 364–389. Springer, 1986. Lecture Notes in Computer Science, Volume 240.
- [55] Donald Sannella and Andrzej Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Science*, 34:150–178, 1987. Earlier version in *Proceedings, Colloquium on Trees in Algebra and Programming*, Lecture Notes in Computer Science, Volume 185, Springer, 1985.
- [56] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Control*, 76:165–210, 1988. Earlier version in *Proceedings, International Symposium on the Semantics of Data Types*, Lecture Notes in Computer Science, Volume 173, Springer, 1985.
- [57] Petros Stefanias. Chartering some institutions, 1993. MSc Thesis, Programming Research Group, Oxford University.
- [58] Andrzej Tarlecki. Free constructions in algebraic institutions. In M.P. Chytil and V. Koubek, editors, *Proceedings, International Symposium on Mathematical Foundations of Computer Science*, pages 526–534. Springer, 1984. Lecture Notes in Computer Science, Volume 176; extended version, University of Edinburgh, Computer Science Department, Report CSR-149-83.
- [59] Andrzej Tarlecki. Bits and pieces of the theory of institutions. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Proceedings, Summer Workshop on Category Theory and Computer Programming*, pages 334–360. Springer, 1986. Lecture Notes in Computer Science, Volume 240.
- [60] Andrzej Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37:269–304, 1986. Preliminary version, University of Edinburgh, Computer Science Department, Report CSR-165-84, 1984.
- [61] Andrzej Tarlecki. Quasi-varieties in abstract algebraic institutions. *Journal of Computer and System Sciences*, 33(3):333–360, 1986. Original version, University of Edinburgh, Report CSR-173-84.
- [62] 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. Also, Monograph PRG-77, August 1989, Programming Research Group, Oxford University.
- [63] Alfred Tarski. The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47, 1944.
- [64] Władysław Turski and Thomas Maibaum. *The Specification of Computer Programs*. Addison-Wesley, 1987.
- [65] Paulo Veloso and Sheila Veloso. Some remarks on conservative extensions: A socratic dialogue. *Bulletin of the European Association for Theoretical Computer Science*, 43:189–198, February 1991.

- [66] Niklaus Wirth. *Programming in Modula-2*. Springer, fourth edition, 1988.
- [67] Keitaro Yukawa. The untyped lambda calculus as a logical programming language, 1990. City University of New York.

Contents

1	Introduction	1
1.1	Some History	3
1.2	Relation with Type Theoretic Approaches	4
1.3	Summary of Results	4
1.4	Prerequisites	5
1.5	Acknowledgements	5
2	Basic Concepts	5
2.1	Institutions	6
2.2	Factorisations and Inclusions	10
2.3	Exactness	15
2.4	π -Institutions	17
3	Basic Module Algebra	19
3.1	Sum	19
3.2	Renaming	20
3.3	Information Hiding	21
3.4	Module Expressions	21
4	Distributive Laws	22
4.1	Distributive Laws and Interpolation	23
4.2	The Weak Distributive Law	26
4.3	The Distributive Law and Conservative Extensions	28
4.4	Summary Concerning the Distributive Laws	28
4.5	The Algebra of Model Classes	29
5	Pushouts of Extensions	31
5.1	Conservative Extensions	32
5.2	Persistent Extensions	33
5.3	Conservative for Initiality	34
6	Conclusions	34