Uniformity, Interpolation and Module specification in a Development Workspace

T. Dimitrakos and T.S.E. Maibaum

Imperial College, 180 Queen's Gate, London SW7 2BZ, U.K.

Abstract. Interpolation and Schematic Reasoning are shown to underlie critical and somewhat complementary aspects of designing and (syntactically) manipulating specification modules. In addition, the presence of a *Uniform* presentation of *interpolants* facilitates the specification of modules. Also, the ability to encapsulate and manipulate *Uniform Schemata* may assist us in reasoning with (abstractions of) hidden data. Unfortunately, most formalisms that have been used in fundamental approaches to software engineering *lack* uniform interpolation and *do not* directly *support* schematic reasoning. This paper reveals the critical role of uniform interpolants and uniform schemata from the perspective of modularity, and quotes a general construction indicating that a potentially large class of calculi can be extended conservatively so that a uniform presentation of the critical interpolants becomes available and the manipulation of uniform schemata is supported.

1 Introduction

There is a well established relation between interpolation [7, 25, 40, 2] and modularity properties of refinement [26, 27, 53, 50, 52, 51, 15, 14, 13] and databases [29]. Also, some operations of module algebras [5] are linked directly with a "splitting" version of interpolation ([37] discussing an earlier version of [5]). The behaviour of the language restriction operator [14, 13](also "information hiding"operator in [9]) is associated with an interpolation property of the specification formalism. In addition, the presence of uniform interpolants facilitates formal reasoning and syntactic manipulations in all the above cases, as explained in [13] and [14]. On the other hand, *uniform schemata*, ie., schemata whose application conserves consequence through language expansions (as the first order equality schema, the first order induction schema, and the classical Hilbert-style axiomatisations do), can be used for encapsulating general properties of abstractions of the "hidden" data. Unfortunately, many logics that have been used in work on refinement or databases *lack* the desirable interpolation properties, and schematic reasoning is often exogenous and it is usually treated by means of purpose-built procedures or tactics that depend intrinsically on the application. To compensate for the absence of modularity, several groups of researchers have proposed techniques to restrict these logics to fragments that have the desirable modularity properties (eg. interpolation/modularisation). Some of these enterprises have focused on algebraic/categorical aspects of modularity (eg. *persistence* [16]), whereas others have emphasised interpolation (eg. *taming logics* [3, 32]). Our approach is in a sense dual [13]: we seek methods to *expand* a specification formalism *orthogonally*, so that an adequately strong version of the critical interpolation properties is obtained and *uniform schemata* become logical. Some features of these methods reflect the way that theorem provers manipulate *meta*-variables and, thus, a possible interpretation of what is going on when the expansion is performed is as "adding meta-theoretic facilities" [4] to one's favorite specification logic. Of course, our intention is *not* to substitute meta-reasoning, but rather to internalise – in a uniform and well-founded way – some fundamental *meta-logical* statements that associate derivability with linguistic transformations.

The paper is structured as follows:

In section 2 we outline the interpolation properties we are interested in, and in section 3 we emphasise their uniform versions. Some (simplified) illustrative examples regarding interpolation, module presentation and information hiding are presented in paragraph 3.2 from the perspective of uniformity. We also present, in association with these examples, a technique for demonstrating, by means of counterexamples, that a given logic *lacks* a uniform version of interpolation. In fact, many logics that have been used for specification design and refinement have been shown to *lack* a uniform version of interpolation.

In section 4 we explain how derivations are affected by a schematic supposition, and we underline some fundamental differences between a schema and a set of sentences with a common syntactic pattern. Then, we distinguish the class of *uniform* schemata and illustrate how they assist in reasoning with (abstractions of) hidden operators.

In an attempt to compensate for the inability of many logics that have been used in computing to provide a uniform presentation of interpolants and to cope with schematic reasoning, we have studied general methods to expand orthogonally a logical formalism $\mathcal{E}_{Spec}$, so that *uniform* interpolants in derivations between $\mathcal{E}_{Spec}$-sentences become available and reasoning with uniform schemata is supported. (The skeleton of such an expansion method has been outlined in [14], described in more detail in [11] and analysed in [13].) The intuition behind the expansion method, together with some salient characteristics of the notions of consequence involved, are quoted in section 5. Then, the previously presented illustrative examples are revisited, emphasising on how this expansion suffices to provide a uniform presentation of the critical interpolants, and a precise encapsulation of the essential constituents of (uniform) schematic reasoning. In the context of this paper the focus is on extending (finitary) propositional and first order languages. The choice of these languages has been made on the basis of familiarity; as is explained in [13], similar methods are applicable for larger classes of logics.

2 Preliminaries

An *Entailment System* [30] $\mathcal{E} = \langle \mathcal{S}\mathbf{ign}, \mathbf{gram}, \vdash^{\mathcal{E}} \rangle$ (also called a π-*Institution* in the chronologically earlier paper [18] and in the more recent revision of [20]) provides an abstract presentation of logical consequence, consisting of

(1) a category $\mathcal{S}\mathbf{ign}$ of signatures;
(2) a functor $\mathbf{gram}{:}\mathcal{S}\mathbf{ign} \to \mathcal{S}\mathbf{et}$, that assigns to each signature $\mathbf{\Sigma}$ the set of sentences/well-formed-formulae built over $\mathbf{\Sigma}$;
(3) a $\mathcal{S}\mathbf{ign}$-indexed family of binary relations (entailment) $\vdash^{\mathcal{L}}$, such that, $\vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \subseteq 2^{\mathbf{gram}(\mathbf{\Sigma})} \times \mathbf{gram}(\mathbf{\Sigma})$ is ***reflexive***, ***monotonic***, ***transitive***, and ***stable under translation*** [1];

The tuple $\mathcal{G}[\mathcal{L}] = \langle \mathcal{S}\mathbf{ign}, \mathbf{gram} \rangle$ is the *Grammar* $\mathcal{G}[\mathcal{E}]$, which presents the (category of) languages in $\mathcal{E}$. In addition, $\mathcal{E}$ is called *compact* iff whenever $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$, there exists a *finite* $A \subseteq \Gamma$ such that $A \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$.

As it is explained in [13] (also noted in [30]), this presentation of logical consequence is independent of the means by which it has been defined (eg. proof-calculus, satisfaction system, forcing, etc.). The usefulness of the *Entailment Systems* framework stems from its power of abstraction. By analysing properties and describing development frameworks by means of *Entailment Systems*, one gains generality: Every concrete development framework that defines the same notion of logical consequence is a candidate for being used in applications[2]. Now let us consider some encapsulations of interpolation properties in the context of *Entailment Systems*:

An *Entailment System* $\mathcal{E}$ possesses Craig-Robinson Interpolation (CRI) iff for every pushout diagram $\mathcal{D}$ in $\mathcal{S}\mathbf{ign}$, as depicted to the right, and every $\mathbf{A} \subseteq \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{A}})$, $\mathbf{B} \subseteq \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{B}})$, $\varphi \in \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{B}})$, such that $\mathbf{A}^{i'} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{C}}} \varphi^{e'}$, there is a set $\mathbf{I}_{(\mathbf{A},\mathbf{B},\varphi,\mathcal{D})} \subseteq \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{R}})$ of *interpolants* such that

$$\begin{tikzcd}
\mathbf{\Sigma}_{\mathsf{A}} \arrow[r, dotted, "i'"] & \mathbf{\Sigma}_{\mathsf{C}} \\
\mathbf{\Sigma}_{\mathsf{R}} \arrow[u, "e"] \arrow[r, "i"'] \arrow[ur, phantom, "\mathcal{D}"] & \mathbf{\Sigma}_{\mathsf{B}} \arrow[u, dotted, "e'"']
\end{tikzcd}$$

1. $\mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{A}}} \mathbf{I}^{e}_{(\mathbf{A},\mathbf{B},\varphi,\mathcal{D})}$,
2. $\mathbf{I}^{i}_{(\mathbf{A},\varnothing,\varphi,\mathcal{D})} \cup \mathbf{B} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{B}}} \varphi$.

where for each translation $i{:}\mathbf{\Sigma}_1 \to \mathbf{\Sigma}_2$ and $\delta \in \mathbf{gram}(\mathbf{\Sigma}_1)$ the writing of $\delta^i \in \mathbf{gram}(\mathbf{\Sigma}_2)$ denotes the image of the formula δ under the translation induced by i. Analogously, if $\Delta \subseteq \mathbf{gram}(\mathbf{\Sigma}_1)$ then $\Delta^i = \{\delta^i : \delta \in \Delta\}$. Finally, $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \Delta$ for $\Gamma, \Delta \subseteq \mathbf{gram}(\mathbf{\Sigma})$ mean $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \delta$ for each $\delta \in \Delta$. Notice the use of the *pushout* construction to capture the idea that the interpolants are in a "shared" language, which is required for stating the property (as in [46]). If $\mathcal{E}$ is *compact* then, for each φ, the set of interpolants is finite. If there is also a deduction theorem for $\mathcal{E}$ then the set of interpolants $\mathbf{I}_{(\mathbf{A},\mathbf{B},\varphi,\mathcal{D})}$ can be viewed independently of $\mathbf{B}$. Finally,

[1] That is, $\{\varphi\} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$; if $\Gamma \subseteq \Delta$ and $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$ then $\Delta \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$; if $\Delta \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$ and $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \delta$, for all $\delta \in \Delta$, the $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}} \varphi$; and for every $i{:}\mathbf{\Sigma}_1 \to \mathbf{\Sigma}_2$ in $\mathcal{S}\mathbf{ign}$, if $\Gamma \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_1} \varphi$ the $\mathbf{gram}(i)(\Gamma) \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_2} \mathbf{gram}(i)(\varphi)$.

[2] Of course, the effectiveness of the concrete framework depends heavily on the means used for defining this logical consequence and some important issues involved in the mechanisation may not appear at the level of *Entailment Systems*.

if $\mathcal{E}$ is a *weakly structural* π-Institution then some *structural axioms* on $\mathcal{D}$ need to be taken into account. (See [20] and [13].)

$\mathcal{E}$ possesses Craig Interpolation (CI) when the above defined property holds (at least) for $\mathbf{B} = \varnothing$. For calculi on propositional and predicate languages, CI coincides with the well-known Craig's Interpolation Lemma [7].

A property similar to CRI first appeared explicitly in the literature in [25] where it was established for the intuitionistic propositional calculus (see also [2]). "Maehara Interpolation" – a term now commonly used in sentential logic–and CRI coincide (at least) for calculi on propositional and predicate languages. In the computer science literature the terms "Splitting", "Strong" and "Pushout" Interpolation have been used to designate either precisely the same or similar (ie., equivalent for calculi on propositional and predicate languages) properties as CRI (see [37, 46, 19, 9] and [20] among others). Finally, the term CRI originally appeared at Shoenfield's book [40] and recently reappeared in [51] and [14, 12]. Note that although CRI is generally stronger than CI (the latter requires the property just for $\mathbf{B} = \varnothing$), the two are equivalent for both classical and intuitionistic propositional and first order logics. In fact, they generally seem to collapse for *compact* calculi with a *deduction detachment property.*

CRI is strongly related with the *Modularisation* property [27, 50, 19, 50, 51, 12, 15, 14] which is an important property for specification design and refinement by means of *implementation steps* [47]. The critical role of MP in specification theory has been analysed in [27, 50, 19, 50, 51, 15, 13], and in [12, 14, 13] from the perspective of uniformity. The *Modularisation* property (MP) is captured in the *Entailment Systems* framework as the preservation of conservative extensions under pushouts:

If $e{:}\langle \mathbf{\Sigma}_{\mathsf{R}}, \mathbf{R}\rangle \to \langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A}\rangle$ is a conservative extension and $\mathcal{D}$ is a pushout diagram in the category of theories over an *Entailment System* $\mathcal{E}$ then $e'{:}\langle \mathbf{\Sigma}_{\mathsf{B}}, \mathbf{B}\rangle \to \langle \mathbf{\Sigma}_{\mathsf{C}}, \mathbf{A}^{i'} \cup \mathbf{B}^{e'}\rangle$ is also a conservative extension.

$$\begin{tikzcd}
\langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A}\rangle \arrow[r, dotted, "i'"] & \langle \mathbf{\Sigma}_{\mathsf{C}}, \mathbf{A}^{i'} \cup \mathbf{B}^{e'}\rangle \\
\langle \mathbf{\Sigma}_{\mathsf{R}}, \mathbf{R}\rangle \arrow[u, "e"] \arrow[r, "i"'] \arrow[ur, phantom, "\mathcal{D}"] & \langle \mathbf{\Sigma}_{\mathsf{B}}, \mathbf{B}\rangle \arrow[u, dotted, "e'"']
\end{tikzcd}$$

The strength of the relation between CRI and MP is underlined in the following statement which is known as the "Modularisation theorem":

An *Entailment System* $\mathcal{E}$ possesses CRI ***iff*** $\mathcal{E}$ possesses the MP.

This equivalence was put forward in [27] and proved in detail in [50] and [52] for the particular case of first order logic. A proof for the general case of an *Entailment System* can be found in [13], together with a critically stronger result showing that he equivalence holds for the (grammatical) locus of *all* theories on $\mathcal{D}$ (ie., the universal meta-quantification on $\mathcal{D}$ is shifted outside the equivalence).

3 Uniform interpolants and Module specifications

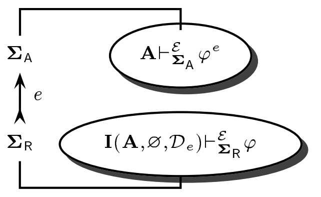
Recall that a module, in specification theory, can be viewed as one (logical) theory A in the employed logic. (See also [5, 9] and [13]). Moreover, if A is the theory of some specification $\langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A} \rangle$ (ie., A has a finite $\mathbf{\Sigma}_{\mathsf{A}}$-axiomatisation $\mathbf{A}$) A is called *specifiable*. In computing, one would clearly prefer all basic modules to be specifiable and all operations on modules to preserve specifiability. Unfortunately, this fails for some simple, intuitively clear, operations based on the "*(sub)module*" of relation[3]: The submodule R of A on $\mathbf{\Sigma}_{\mathsf{R}}$ is given by the *restriction* of the theory A to the sublanguage $\mathbf{gram}(\mathbf{\Sigma}_{\mathsf{R}})$. If A is specifiable by $\langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A} \rangle$ then R is called the $\mathbf{\Sigma}_{\mathsf{R}}$-module of $\langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A} \rangle$. The purpose of this section is define the notion of a *uniform interpolant* and disclose a strong connection between the existence of uniform interpolants and the specifiability of (sub)modules of logical specifications.

Assume that an *Entailment System* $\mathcal{E}$ possesses CRI. Given a pushout diagram $\mathcal{D}$ in the category of signatures, and $\mathbf{A} \subseteq \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{A}})$, $\mathbf{B} \subseteq \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{B}})$ we say that interpolants have a *uniform presentation* iff there is a set of interpolants $\mathbf{I}(\mathbf{A},\mathbf{B},\mathcal{D})$ such that:

(1). $\mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{A}}} \mathbf{I}^{e}(\mathbf{A},\mathbf{B},\mathcal{D})$,
(2). $\mathbf{I}(\mathbf{A},\mathbf{B},\mathcal{D}) \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{R}}} \mathbf{I}(\mathbf{A},\mathbf{B},\varphi,\mathcal{D})$ for every $\varphi \in \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{B}})$, and
(3). $\mathbf{I}(\mathbf{A},\mathbf{B},\mathcal{D})$ is finite whenever $\mathbf{A}$ is finite.
(ie. for each assertion $\alpha \in \mathbf{A}$ there is a finite set of interpolants which is stronger than any other interpolants and independent of the consequence φ)

$\mathcal{E}$ possesses *Uniform Craig-Robinson Interpolation* (UCRI) iff for every $\mathbf{A}$, $\mathbf{B}$ and $\mathcal{D}$ there is a uniform presentation $\mathbf{I}(\mathbf{A},\mathbf{B},\mathcal{D})$ of the Craig-Robinson interpolants. Consequently, $\mathcal{E}$ possesses *Uniform Craig Interpolation* (UCI) iff for every $\mathbf{A}$, $\mathbf{B}$ and $\mathcal{D}$ there is a uniform presentation $\mathbf{I}(\mathbf{A},\varnothing,\mathcal{D})$ of the Craig interpolants. (Alternatively, the requirement of *uniformity* can be directly embodied in the definition of CRI (resp. CI) by strengthening the property as in [15, 14] and [13].)

An immediate consequence of *Uniform* interpolation is that *the restriction* R *of a finitely axiomatisable theory* A *to a sublanguage* $\mathbf{\Sigma}_{\mathsf{R}}$ *has some finite* $\mathbf{\Sigma}_{\mathsf{R}}$-*axiomatisation* (consisting of the uniform interpolants).



$\mathbf{\Sigma}_{\mathsf{A}}$
$\mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{A}}} \varphi^{e}$
e
$\mathbf{\Sigma}_{\mathsf{R}}$
$\mathbf{I}(\mathbf{A},\varnothing,\mathcal{D}_e) \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{R}}} \varphi$

Recall that the theory R of the restriction of A $= \langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A} \rangle$ to a sublanguage $\mathbf{gram}(\mathbf{\Sigma}_{\mathsf{R}})$ is defined as the set of the $\mathbf{\Sigma}_{\mathsf{R}}$-consequences of $\mathbf{A}$, ie., $\mathsf{R} = \{\varphi \in \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{R}}) : \mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{A}}} \varphi\}$. Now if $\mathbf{I}(\mathbf{A},\varnothing,e)$ is the (set of) uniform interpolant(s), and e is the subsignature inclusion $e{:}\mathbf{\Sigma}_{\mathsf{R}} \to \mathbf{\Sigma}_{\mathsf{A}}$ (viewed as pushout of e along $id_{\mathbf{\Sigma}_{\mathsf{R}}}$, identity morphism on $\mathbf{\Sigma}_{\mathsf{R}}$), then a(ny) $\mathbf{\Sigma}_{\mathsf{R}}$-sentence φ belongs to R ***iff*** $\mathbf{I}(\mathbf{A},\varnothing,e) \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{R}}} \varphi$. The conditional is immediate from the definition of uniform interpolation and the converse conditional follows because $\vdash^{\mathcal{E}}$ is *transitive* and *stable under translation*.

[3] Note that modules have been defined in a variety of different ways in programming, specification and automated reasoning. The definition sketched here is a simplification / abstraction of those which are common in specification theory and consistent with the view of the specification design and development as theory manipulation. In any case, most seem to agree on the conservativeness underlying the "*is a (sub)module of*" relation.

An obvious use of the above equivalence (also noted in [14]) is as a method for deriving negative results, ie., for proving that a logic *lacks Uniform* interpolation, it suffices to find a specification A on $\mathbf{\Sigma}_\mathsf{A}$ and a subsignature $\mathbf{\Sigma}_\mathsf{R}$ on which A has a (finitely) unspecifiable restriction. Such a counterexample for first order logic is given by restricting the specification of a *dense linear order* to the language of equality. It is well known (there have been several arguments mostly from game theory) that linear density is not specifiable by means of a finite number of first order sentences in the language of equality. Similar counterexamples can be found easily for equational logic, propositional dynamic logic, infinitary logic ($\mathbf{L}_{\omega_1\omega}$), monadic second order logic, etc.

3.1 Information Hiding

Information Hiding is an important and well-known technique in both conventional programming and formal specification design. Already in the early *70's*, Parnas [34, 33] had emphasised the importance of hiding implementation details within a module. This is accomplished by allowing access to the data representation of a specification only through operations exported by its module. In a somewhat similar spirit, Bergstra and Tucker [1] had shown that *any recursive* $\mathbf{\Sigma}_\mathsf{R}$-algebra can be specified as the $\mathbf{\Sigma}_\mathsf{R}$-restriction of an initial $\mathbf{\Sigma}_\mathsf{A}$-algebra defined by a finite set $\mathbf{A}$ of $\mathbf{\Sigma}_\mathsf{A}$-equations, attempting to compensate for the negative result of [28] that certain $\mathbf{\Sigma}_\mathsf{R}$-algebras *cannot* be specified as the initial $\mathbf{\Sigma}_\mathsf{R}$-algebra of a finite set of $\mathbf{\Sigma}_\mathsf{R}$-equations. In many calculi that are currently used in computing, there are interesting $\mathbf{\Sigma}_\mathsf{R}$-theories which *lack* a finite presentation on $\mathbf{gram}(\mathbf{\Sigma}_\mathsf{R})$ and can be seen only as $\mathbf{\Sigma}_\mathsf{R}$-restrictions of some specification $\langle \mathbf{\Sigma}_\mathsf{A}, \mathbf{A} \rangle$ which includes both *visible* and *hidden* features.

It easily follows from our preceding analysis that if a calculus (presented by means of an *Entailment System* $\mathcal{E}$) possesses UCRI, then the above problems of specifying modules reduces to the problem of deriving uniform interpolants: all (sub)modules based on a specification $\langle \mathbf{\Sigma}_\mathsf{A}, \mathbf{A} \rangle$ are (directly) specifiable. An obvious finite axiomatisation is given by the corresponding set of uniform interpolants. Furthermore, if $\mathcal{E}$ possesses uniform interpolation (UCI also suffices) *locally* over e, then the $\mathbf{\Sigma}_\mathsf{R}$-module of $\langle \mathbf{\Sigma}_\mathsf{A}, \mathbf{A} \rangle$ is specifiable and $\langle \mathbf{\Sigma}_\mathsf{R}, \mathbf{I}_{(\mathbf{A},\varnothing,e)} \rangle$ is a specification of this module: If $\mathbf{\Sigma}_\mathsf{R}$ is a subsignature of $\mathbf{\Sigma}_\mathsf{A}$, denoted by $e{:}\mathbf{\Sigma}_\mathsf{R} \rightarrow \mathbf{\Sigma}_\mathsf{A}$, then a specification $\mathsf{R} = \langle \mathbf{\Sigma}_\mathsf{R}, \mathbf{R} \rangle$ is a $\mathbf{\Sigma}_\mathsf{R}$-*module* of $\mathsf{A} = \langle \mathbf{\Sigma}_\mathsf{A}, \mathbf{A} \rangle$ ***iff*** $\mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_\mathsf{A}} \mathbf{R}^e$ and $\mathbf{R} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_\mathsf{R}} \mathbf{I}_{(\mathbf{A},\varnothing,e)}$.

$$\begin{tikzcd}
\mathbf{R} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_\mathsf{R}} \mathbf{I}(\mathbf{A},\varnothing,e) \arrow[r, dash] & \mathbf{\Sigma}_\mathsf{R} \arrow[r, tail, "e"] & \mathbf{\Sigma}_\mathsf{A} \arrow[r, dash] & \mathbf{A} \vdash^{\mathcal{E}}_{\mathbf{\Sigma}_\mathsf{A}} \mathbf{R}^e
\end{tikzcd}$$

Generalising, *every finitely axiomatisable theory on* $\mathbf{\Sigma}_\mathsf{A}$ *has a finitely axiomatisable restriction on* $\mathbf{\Sigma}_\mathsf{R}$ *(wrt. e)* ***iff*** UCI *holds locally on e.*

The following examples may assist the reader in understanding the role of interpolants in a derivation in relation to uniformity. For simplicity we use unsorted grammar and we assume familiarity with the calculi of (classical) propositional and first order logics, denoted by $\mathcal{CPC}$ and $\mathcal{CFOC}$ respectively.

3.2 Examples

Let $\mathbf{\Sigma}_{\mathsf{A}}$ present a propositional alphabet including the propositional symbols p, r and q and let $e{:}\mathbf{\Sigma}_{\mathsf{R}}{\to}\mathbf{\Sigma}_{\mathsf{A}}$ denote the inclusion of the subalphabet $\mathbf{\Sigma}_{\mathsf{R}}$ in $\mathbf{\Sigma}_{\mathsf{A}}$ such that $\mathbf{\Sigma}_{\mathsf{R}}$ excludes q (ie., q is the "hidden" proposition). Consider an arbitrary formula ψ in the propositional language of $\mathbf{\Sigma}_{\mathsf{B}}$, and let $\mathbf{D} = \boxed{\mathbf{A}\vdash^{\mathcal{CPC}}_{\mathbf{\Sigma}_{\mathsf{A}}} \psi^{e}}$ be a derivation in $\mathcal{CPC}$ over the language of $\mathbf{\Sigma}_{\mathsf{A}}$. If $\mathbf{A} = \{\mathrm{p}\wedge\mathrm{q}\}$, then the sentence p is a $\mathbf{\Sigma}_{\mathsf{R}}$-interpolant for $\mathbf{D}$. Under the same assumptions, if $\mathbf{A} = \{(\mathrm{p}\wedge\mathrm{q})\vee\mathrm{r}\}$, then sentence $p\vee r$ is an $\mathbf{\Sigma}_{\mathsf{R}}$-interpolant for $\mathbf{D}$, and if $\mathbf{A} = \{\mathrm{p}\wedge(\mathrm{q}\vee\mathrm{r})$ then the sentence p is a $\mathbf{\Sigma}_{\mathsf{R}}$-interpolant for $\mathbf{D}$. All these interpolants are uniform: they depend on the logical structure of the assertion and on the *language* of the derivation; they are independent of the logical structure of the consequence ψ. Hence, $\{p\}$ axiomatises the $\{p\}$-module of $\langle\{p,q\},\{p\wedge q\}\rangle$. Also $\{p\}$ axiomatises the $\{p,r\}$-module of $\langle\{p,q,r\},\{p\wedge(q\vee r)\}\rangle$, and, finally, $\{p\vee r\}$ axiomatises the $\{p,r\}$-module of $\langle\{p,q,r\},\{(p\wedge q)\vee r\}\rangle$. In fact, the classical propositional calculus $\mathcal{CPC}$ possesses UCRI (thus UCI) and therefore every module a propositional specification is *directly* specifiable.

Let $\mathbf{\Sigma}_{\mathsf{A}}$ be a first order alphabet which includes a function symbol f and let $e{:}\mathbf{\Sigma}_{\mathsf{R}}{\to}\mathbf{\Sigma}_{\mathsf{A}}$ $\mathbf{\Sigma}_{\mathsf{R}}$, with e denoting the inclusion of the subalphabet $\mathbf{\Sigma}_{\mathsf{R}}$ of $\mathbf{\Sigma}_{\mathsf{A}}$ excluding f. Let $\mathbf{A}$ be characterised by the sentence $\forall(x)\,(I\,(x)\to O(x,f(x)))$, (which may present the input-output behaviour of a functional program) in the first order language of $\mathbf{\Sigma}_{\mathsf{A}}$). The sentence $\vartheta_{\mathbf{A}} = \forall(x)\exists(z)\,(I\,(x)\to O\,(x,z))$ is an $\mathbf{\Sigma}_{\mathsf{R}}$-interpolant of $\mathbf{A}\vdash^{\mathcal{CFOC}}_{\mathbf{\Sigma}_{\mathsf{A}}} \psi^{e}$, where f does not appear in ψ, regardless of the logical structure of the consequence ψ. In this sense, $\langle\mathbf{\Sigma}_{\mathsf{R}},\vartheta_{\mathbf{A}}\rangle$ is a presentation of the $\mathbf{\Sigma}_{\mathsf{R}}$-module of the specification $\langle\mathbf{\Sigma}_{\mathsf{A}},\{\forall(x)\,(I\,(x)\to O(x,f(x)))\}\rangle$. Again, the interpolant is uniform: it does not depend on the logical structure of the consequence. In general, one can show [13] (extending [54]) that in first order (classical/intuitionistic) logic all derivations of the form $\varphi[x,f(x)]\vdash\psi$, where f does not appear in ψ, have a uniform interpolant [4].

There are derivations, though, in first order logic where the form of the interpolant depends on the logical structure of both the assertions and the consequence. Let $\mathbf{\Sigma}_{\mathsf{A}}$ also include another function symbol g which is not in the subsignature $\mathbf{\Sigma}_{\mathsf{R}}$, and $\mathbf{A}$ consisting of the sentence $\forall(x,y)\,(I(x,y)\to O\,(x,f(x),y,g\,(y)))$. Then the interpolant depend on both the assertion and the consequence. This is partially due to the fact that one may obtain syntactically different and logically non-equivalent sentences when abstracting the operators f and g from the (conjunction of) the assertions (eg. $\forall(x)\exists(z)\forall(y)\exists(w)I(x,y)\to\ O(x,z,y,w)$ and $\forall(y)\exists(w)\forall(x)\exists(z)I(x,y)\to\ O(x,z,y,w)$), one of which may be used for deriving the consequence ψ depending on ψ's logical structure. Also, *no combination of such sentences*, in a first order syntax, is able to capture the fact that each of

[4] In fact, $f(x)$ may denote syntactically identical occurrences of either a function or a predicate. See [10] and [13] for details.

z, w depends only on one of x, y and is independent of the other. A (highly non-deterministic) algorithm for generating (non-uniform) first order interpolants for a derivation $\mathbf{D}$ using Unskolemization and Resolution is presented in [6].

Another example of first order derivations where the form of the interpolant depends on the logical structure of both the assertions and the consequence is obtained by considering the finite axiomatisation of a *Dense Linear Order* in a language $\mathbf{\Sigma}_{\mathsf{A}}$ incorporating an order symbol $<$ (and, of course, logical equality):

anti-reflexivity $\forall(x)\, \neg(x < x)$
transitivity $\forall(x, y, z)\; x < y \;\wedge\; y < z \rightarrow x < z$
totality $\forall(x, y)\; x < y \;\vee\; y < x \;\vee\; x = y$
bottom element $\forall(x)\, \neg(x < 0)$
density $\forall(x, y)\exists(w)\; x < y \rightarrow (x < z \wedge z < w)$

As already mentioned in this paper, it is impossible to obtain a finite description of (the consequences of) linear density in the language of equality. Therefore, first order derivations of equational sentences from the (conjunction of) the above axioms depend on both the assertions and the consequence. As explained in [13], similar counterexamples hold for infinitary logic ($\mathbf{L}_{\omega_1\omega}$) and monadic second order logic. Counterexamples for equational logic can be produced based on [28] and [1] (read also [31]). Calculi that *possess* a *uniform version of interpolation* include classical propositional logic, (Heyting's) intuitionistic propositional logic [35], Löb's logic (GL) [39], Gregorczyk's logic ($\mathsf{S4Grz}$) [55], polymodal (Hennessy-Milner) logic and μ-calculus [8], and classical and intuitionistic second order predicate logics with predicate variables of *all* arities.

4 Modules and Uniform schemata

According to *Webster's Dictionary* (1913), a *schema* is "*an outline or image universally applicable to a general conception, under which it is likely to be presented to the mind*". This perception of schematic representation originates in Kantian philosophy and has been exploited in "The Critique of Pure Reason" [24] (see also [36]). According to a Dictionary of Mathematics, a *schema* in mathematical logic is "*a method of representing a possibly infinite number of wffs, of some object language by using metalinguistic expressions that take object language as substitution instances...*". But a schema is not a pure metalinguistic expression, in the sense that symbols of a "base" object alphabet may appear as fixed constituents of it. A schema may be conceived as a linguistic meta-form indexed by the (category of) languages that include the language of a "base" signature $\mathbf{\Sigma}_{\mathsf{R}}$, such that $\mathbf{\Sigma}_{\mathsf{R}}$ is constituted by the ("object") symbols that appear fixed in the schema. symbols appear fixed in it. In that sense, a schema is both denotationally and operationally different from an arbitrary set of commonly structured $\mathbf{\Sigma}$-sentences as we will elaborate on in the sequel. Consider as an example the familiar first order induction schema:

(FOInd) *For every predicate* φ [in all languages extending $\mathbf{\Sigma}_{\mathbb{N}} = \langle\{0, succ\}, \{=\}\rangle$]
$$\varphi(0) \wedge (\forall(x)\varphi(x) \rightarrow \varphi\,(s\,(x))) \rightarrow \forall(x)\varphi(x)$$

The fundamental "operational" difference between the supposition of the above schema (FOInd) and its $\mathbf{\Sigma}_{\mathbb{N}}$ instance $\mathbf{Ind}_{\mathbb{N}} = \{\varphi(0) \wedge (\forall(x)\varphi(x) \rightarrow \varphi\,(s\,(x))) \rightarrow \forall(x)\varphi(x)\varphi \in \boldsymbol{Frm}(\mathbf{\Sigma}_{\mathbb{N}})\}$ is that the schema *re-populates* the set of assertions for each language expansion, whereas **Ind** (on $\mathbf{\Sigma}_{\mathbb{N}}$ *does not*.

An example, first mooted in Enderton's book [17] (and analysed from different perspectives in [49] and [13]), may assist in a further clarification of the above. The following set of sentences is a *finite* axiomatisation of a *complete* (maximally consistent) and *effectively decidable* theory Nat on the first order language of $\mathbf{\Sigma}_{\mathsf{Nat}} = \langle\{0, succ\}, \{<\}\rangle$ which has $\mathbb{N}$ (the natural numbers) as one model:

$$\textit{discrete linear order: } \begin{Bmatrix} \forall x \forall y\,(x < y \rightarrow \neg y < x) \\ \forall x \forall y\,(x < y \;\vee\; x = y \;\vee\; y < x) \\ \forall x \forall y\,((x < y \;\wedge\; y < z) \rightarrow x < z) \end{Bmatrix}$$

immediate successor: $\forall x \forall y\,(x < succ(y) \rightarrow \neg y < x)$
initial element: $\forall x\,(\neg succ(x) = 0)$
(abstracted) predecessor: $\forall(y)\exists(x)\,(y \neq 0 \rightarrow y = succ(x))$

These axioms specify a discrete linear order with immediate successor and an initial element in a language $\mathbf{\Sigma}_{\mathsf{Nat}}$ which is *not expressive enough* to interpret Peano arithmetic, and therefore neither the completeness nor the (effective) decidability of the presented theory contradict Gödel's incompleteness theorems [21]. Also note that there is no assumption of induction over $\mathbf{\Sigma}_{\mathsf{Nat}}$. Though, all inductive statements in the language of $\mathbf{\Sigma}_{\mathsf{Nat}}$, ie. $\mathbf{Ind}_{\mathsf{Nat}} = \{\varphi(0) \wedge (\forall(x)\varphi(x) \rightarrow \varphi\,(s\,(x))) \rightarrow \forall(x)\varphi(x) : \varphi \in \mathbf{gram}(\mathbf{\Sigma}_{\mathsf{Nat}})\}$ are derivable, as the theory Nat is complete and has $\mathbb{N}$ as a model. Hence, the (*meta*-)property described by *(*FOInd*)* is approximated by Nat. But it is not totally possessed. If $\mathbf{\Sigma}_{\mathsf{Nat}}$ is extended to $\mathbf{\Sigma}_{\mathsf{NatPlus}}$ by appending a binary operator $+$ to $\mathbf{\Sigma}_{\mathsf{Nat}}$, induction instances involving the operator $+$, like $(\,\varphi(0)\,(\wedge\forall(x)\varphi(x) \rightarrow \varphi\,(s\,(x))) \rightarrow \forall(x)\varphi(x)\,)\,[\varphi \mapsto (\forall(y)y < succ(x) + y\,)\,]$ are neither provided nor derivable in $\mathsf{NatPlus}$. As a result, when Nat is (conservatively) extended to $\mathsf{NatPlus}$ by appending the following inductive definition of an addition operator $+$, *(Plus0)*$\forall(y)\,(0 + y = y)$ and *(Plus1)* $\forall(x)\forall(y)\,(succ(x) + y = succ(x + y))$, one *unable* to derive $\forall(x)\forall(y)\; y < succ(x) + y$ as a theorem, as one might expect. (See [13] for details and [49] for some other related comments.) Concluding, schemata are meta-statements expressing properties that hold globally (for all expansions of the base language). As such, they cannot be identified with a(ny) particular set of sentences in the language of some specific alphabet.

A schema USch is called *uniform* iff its instantiations conserve (logical) consequence along language expansions. That is, if S and T are theories generated by instantiating USch on $\mathbf{\Sigma}_{\mathsf{S}}$ and $\mathbf{\Sigma}_{\mathsf{T}}$ respectively, and $\mathbf{\Sigma}_{\mathsf{S}}$ is a subsignature of $\mathbf{\Sigma}_{\mathsf{T}}$, then T is a *conservative* over S, ie., the S is the $\mathbf{\Sigma}_{\mathsf{S}}$-module of T. The above stated FOInd, the equality schema and the classical Hilbert-style axiomatisations are examples, from first order logic, of familiar uniform schemata.

With the notion of a uniform schema at hand, we are able to note a fundamental aspect of the interrelation schemata and modules: Uniform schemata assist in encapsulating and reasoning with general properties of data that *can* *be* hidden, based on the accessible information. By assuming (resp. validating) a uniform schema over a module R, one imposes the supposition (resp. verifies) that a property holds in *all* different contexts that enclose R as a module.

In order to clarify the above by means of a concrete example, we have re-examined the *strong distributive law* as it was presented in [9]. An *Entailment System* $\mathcal{E}$ possesses the *strong distributive law* for module sums in $\mathcal{E}$ iff for every pair of specifications $\mathsf{A} = \langle \mathbf{\Sigma}_{\mathsf{A}}, \mathbf{A} \rangle$, $\mathsf{B} = \langle \mathbf{\Sigma}_{\mathsf{B}}, \mathbf{B} \rangle$, and every common sub-signature $\mathbf{\Sigma}_{\mathsf{R}}$ of $\mathbf{\Sigma}_{\mathsf{A}}$ and $\mathbf{\Sigma}_{\mathsf{B}}$, the theory $[\mathsf{A}+\mathsf{B}]_{\mathsf{R}}$ of the restriction of the sum $\mathsf{A}+\mathsf{B}$ to $\mathbf{gram}(\mathbf{\Sigma}_{\mathsf{R}})$ coincides with the sum $[\mathsf{A}]_{\mathsf{R}} + [\mathsf{B}]_{\mathsf{R}}$ of the corresponding $\mathbf{\Sigma}_{\mathsf{R}}$-restrictions (ie., the restriction operator distributes over the sum: the $\mathbf{\Sigma}_{\mathsf{R}}$-module of the sum equals the sum of the $\mathbf{\Sigma}_{\mathsf{R}}$-modules). As explained in [9], the *strong distributive law* does not hold for most calculi used in fundamental approaches to software engineering: The theory of the $\mathbf{\Sigma}_{\mathsf{R}}$-module of the sum $\langle \mathbf{\Sigma}_{\mathsf{A+B}}, \mathbf{A}+\mathbf{B} \rangle$ may be richer (ie., properly include) the theory of the sum $[\mathsf{A}]_{\mathsf{R}} + [\mathsf{B}]_{\mathsf{R}}$ of the corresponding modules, due to sharing/side-effect interaction of the "hidden" data, as is demonstrated by the following example: Let A be $\langle \{0,1,2\}, \{0=1\} \rangle$, B be $\langle \{0,1,2\}, \{0=2\} \rangle$ and let $\mathbf{\Sigma}_{\mathsf{R}}$={1,2}. The theory $\mathsf{A}+\mathsf{B}$ is presented by $\langle \{0,1,2\}, \{0=1, 0=2\} \rangle$ and therefore $[\mathsf{A}+\mathsf{B}]_{\mathsf{R}}$ possesses the sentence $1=2$ as theorem, whereas $[\mathsf{A}]_{\mathsf{R}}$ and $[\mathsf{B}]_{\mathsf{R}}$ are both presented by $\langle \{1,2\}, \varnothing \rangle$, and therefore the theory $[\mathsf{A}]_{\mathsf{R}} + [\mathsf{B}]_{\mathsf{R}}$ is also presented by $\langle \{1,2\}, \varnothing \rangle$ which does not possess $1=2$ as a theorem. Clearly, $[\mathsf{A}+\mathsf{B}]_{\mathsf{R}}$ *includes* $[\mathsf{A}]_{\mathsf{R}} + [\mathsf{B}]_{\mathsf{R}}$ as expected.

Let us note now, how reasoning by means of *uniform schemata* can assist in alleviating the absence of direct knowledge of a possible interaction between "hidden" data: First of all, it cannot (actually it *should not*) eliminate the "problem". No method of conservatively extending an *Entailment System* $\mathcal{E}$ can attribute to $\mathcal{E}$ itself the strong distributivity the $\mathcal{E}$ lacks. In fact, attributing $\mathcal{E}$ strong distributivity may not be even desirable, since it disallows interaction between hidden "data" which is something natural and, depending on the application, may be essential. Reasoning with uniform schemata offers critical assistance in predicting or detecting the possible consequences of sharing or interaction of *potentially "hidden"* data. The critical uniform schema in the above mentioned example is given by *"for all (constant) terms t in all languages that include* $\mathbf{\Sigma}_{\mathsf{R}}$ $=$ *{0,1}, if* $t=1$ *and* $t=2$ *then* $1=2$*"*, which is *valid* for *Entailment System*s with (logical) equality, and it is able to detect sharing/interactions of a similar nature to those that cause the absence of strong distributivity. The latter is obtained by the ability to incorporate (meta)linguistic abstractions of the operators that may appear in an arbitrary context that embodies the given module. And, in that sense, it is presentable by means of *the accessible information* only. The importance of uniformity resides in this case in guaranteeing that the accessible data describe modules of *all* the contexts under consideration. Had the uniformity requirement on the schema been abandoned, one would account all contexts that extend the accessible data, rather than those that embody these data as a module.

5 The *Development Workspace* approach

The development of a general method to expand *orthogonally* the logical consequence of a specification formalism so that a *uniform* presentation of the *critical* interpolants and a precise encapsulation of the essential logical structure of *uniform* schemata become available has been outlined in [14], described in [11] and analysed in [13]. This expansion is presented by means of a *Subentailment System* [30] called a *Development Workspace* [13], and seems suitable as *(i)* a (theoretical) framework where precise formal encapsulations of the desired modularity properties can be studied, *(ii)* an abstract construction providing insights and inspiration for extending concrete calculi and satisfaction systems, and *(iii)* a classifier for detecting the adequacy of extensions which attempt to "fix" modularity problems of concrete calculi and satisfaction systems. A precise description of the structure of a *Development Workspace* cannot be presented herein, for the available space is limited and since the purpose of this paper is to illustrate the potential usefulness of this method (rather than presenting the method in detail). The reader whose interest in this approach is motivated is encouraged to consult the cited references for further details. For the purposes of this section, we put emphasis on the underlying grammatical expansion and some of the salient characteristics of the interrelation between the logical consequence at the source (*"specification"*) and the extension (*"development"*) level. The basic idea behind the construction of a *Development Workspace* $\langle \mathcal{E}_{Spec}, \mathcal{J} \rangle$ on a specification formalism $\mathcal{E}_{Spec}$, is just to *internalise* – in a uniform and well-founded way – some fundamental *meta-logical* statements that associate derivability with linguistic transformations.

A *Development Workspace* $\langle \mathcal{E}_{Spec}, \mathcal{J} \rangle$ embodies an expansion $\mathcal{J}{:}\mathcal{E}_{Spec} \to \mathcal{E}_{Dev}$ of $\mathcal{E}_{Spec}$ to $\mathcal{E}_{Dev}$ so that the grammar of $\mathcal{E}_{Dev}$ differs from $\mathcal{E}_{Spec}$ only with respect to the logical operators: The main feature of the expansion is the augmentation of the specification syntax with new *"development"* variables $\varkappa^n$ that abstract n-ary atoms and with new operators $\boxed{\forall}$ and $\Diamond\!\!\!\!\exists$ to bind these variables. One possible interpretation of the new operators is as term/predicate quantifiers that are *relativised* on the (codomain of) language expansions. In this sense, an $\mathbf{\Sigma}_{\mathsf{R}}$-sentence $\boxed{\forall}\varkappa^n\varphi$, of $\mathcal{E}_{Dev}$, where $\boxed{\forall}$, $\Diamond\!\!\!\!\exists$ do not occur in φ is interpreted as

> *"for all signatures* $\mathbf{\Sigma}_{\mathsf{A}}$ *that include* $\mathbf{\Sigma}_{\mathsf{R}}$ *and all* n*-ary operations* χ *in* $\mathbf{gram}_{\mathcal{E}_{Spec}}(\mathbf{\Sigma}_{\mathsf{A}})$*,* $\varphi[\varkappa^n \mapsto \chi]$*"*

and, $\Diamond\!\!\!\!\exists\varkappa^n\varphi$ is analogously interpreted as

> *"for some signature* $\mathbf{\Sigma}_{\mathsf{A}}$ *that includes* $\mathbf{\Sigma}_{\mathsf{R}}$ *and some* n*-ary operation* χ *in* $\mathbf{gram}_{\mathcal{E}_{Spec}}(\mathbf{\Sigma}_{\mathsf{A}})$*,* $\varphi[\varkappa^n \mapsto \chi]$*"*

The specification *Grammar* $\mathcal{G}[\mathcal{E}_{Spec}]$ is expanded to the development *Grammar* $\mathcal{G}[\mathcal{E}_{Dev}]$ as follows:

(a) the category of extra-logical alphabets (signatures) remains intact;
(b) an unlimited collections of new n-ary *"development" variables* ($\varkappa_i^n$) are added , for each arity n, and used as grammatical abstractions ("placeholders") of $\mathcal{E}_{Spec}$-expressions;

(c) two new logical operations $\boxed{\forall}$, $\Diamond\!\!\exists$ to bind the new variables are added;

(d) a lambda abstraction operator λ to bind free *"specification"* variables is added: For every formula $\overline{\psi}[x_1...x_n]$ in $\mathbf{gram}_{\mathcal{E}_{Dev}}(\mathbf{\Sigma})$ with $x_1...x_n$ free "specification" variables, and such that *neither* $\boxed{\forall}$ *nor* $\Diamond\!\!\exists$ appear in $\overline{\psi}$, the expression $\lambda x_1...x_n.\overline{\psi}$ is an n-ary *"development" term* in $\mathbf{gram}_{\mathcal{E}_{Dev}}(\mathbf{\Sigma})$;

(e) free occurrences of n-ary *development variables* can be substituted by n-ary *development terms* by a "bound" substitution that reduces the redex: $\varkappa^2(t_1,t_2)[\varkappa^2 \mapsto \lambda x_1,x_2.\varphi[x_1,x_2]] \equiv (\varphi\,[x_1,x_2])\,[x_1 \mapsto t_1, x_2 \mapsto t_2] \equiv \varphi(t_1,t_2)$;

(f) all formulae of $\mathbf{gram}_{\mathcal{E}_{Dev}}(\mathbf{\Sigma})$ are generated by *(a)-(e)* above.

In terms of logical consequence, the salient characteristics of a *"Development Workspace"* which are important for module representation and schematic reasoning include the following:

For every signature $\mathbf{\Sigma}$, $\vdash^{\mathcal{E}_{Dev}}$ is *conservative* over $\vdash^{\mathcal{E}_{Spec}}$, ie., for every $\Gamma \subseteq \mathbf{gram}_{\mathcal{E}_{Spec}}(\mathbf{\Sigma})$ and every $\varphi \in \mathbf{gram}_{\mathcal{E}_{Spec}}(\mathbf{\Sigma})$, $\Gamma \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}} \varphi$ iff $\Gamma \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}} \varphi$.

If φ is a sentence in $\mathbf{gram}_{\mathcal{E}_{Spec}}(\mathbf{\Sigma}_\mathsf{A})$ and ς is an n-ary operation of $\mathbf{\Sigma}_\mathsf{A}$, then the $\mathcal{E}_{Dev}$-derivation $\varphi \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_\mathsf{A}} \Diamond\!\!\exists \varkappa^n \varphi[\varsigma \mapsto \varkappa^n]$ is valid.

$\mathcal{E}_{Dev}$ provides a *uniform* presentation of interpolants in derivations between $\mathcal{E}_{Spec}$- sentences, and all these *uniform interpolants* are instances of the same syntactical pattern: A uniform interpolant for a $\mathcal{E}_{Spec}$-derivation $\varphi \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}_\mathsf{A}} \psi$ in particular, is the $\mathcal{E}_{Dev}$-sentence $\Diamond\!\!\exists \varkappa_1^{n_1} ... \Diamond\!\!\exists \varkappa_k^{n_k} \varphi[\varsigma_1 \mapsto \varkappa_1^{n_1}, ..., \varsigma_k \mapsto \varkappa_k^{n_k}]$ where $\{\varsigma_1, ..., \varsigma_k\}$ are the set of $\mathbf{\Sigma}_\mathsf{A}$ symbols that appear in the assertion φ, and *do not* appear in the consequence ψ. The usually hard process of deriving interpolants is thus trivialised. Furthermore, if $\mathcal{E}_{Spec}$ possesses CRI (resp. CI) then $\mathcal{E}_{Dev}$ possesses UCRI (resp. UCI) overall.

On the other hand, if $\boxed{\forall} \varkappa^n \varphi[\varkappa^n]$ is a sentence in $\mathbf{gram}_{\mathcal{E}_{Dev}}(\mathbf{\Sigma}_\mathsf{R})$, the signature $\mathbf{\Sigma}_\mathsf{A}$ includes $\mathbf{\Sigma}_\mathsf{R}$ and ς is an n-ary operation (of the same type as $\varkappa^n$) then the $\mathcal{E}_{Dev}$-derivation $\boxed{\forall} \varkappa^n \varphi \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_\mathsf{A}} \varphi[\varkappa^n \mapsto \lambda\varsigma]$ is valid. Finally, if $\Gamma \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}_\mathsf{R}} \varphi[\varkappa^n \mapsto \lambda\varsigma]$ is a derivation over $\mathbf{\Sigma}_\mathsf{R}$, and for each language $\mathbf{\Sigma}_\mathsf{A}$ that includes $\mathbf{\Sigma}_\mathsf{R}$ and all n-ary operations ς' (of the same type as ς) in $\mathbf{\Sigma}_\mathsf{A}$, $\Gamma \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}_\mathsf{A}} \varphi[\varkappa^n \mapsto \lambda\varsigma']$ then $\Gamma \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_\mathsf{A}} \boxed{\forall} \varkappa^n \varphi[\varkappa^n]$.

Note that the generalisation of the above to a finite set of assertions $\mathbf{A}$ on $\mathbf{\Sigma}_\mathsf{A}$, requires (finite) conjunction – or equivalent – in the $\mathcal{E}_{Spec}$syntax. But this raises no barrier; the addition of (finite) conjunction is always conservative. So, if $\mathcal{E}_{Spec}$ does not possess logical conjunction then a conjunction operator may need to be added before the expansion sketched above is applied.

One of the main contributions of a *Development Workspace* is that $\mathcal{E}_{Dev}$ provides *uniform* interpolants for derivations between $\mathcal{E}_{Spec}$-sentences:

For every pushout diagram $\mathcal{D}$ in $\mathcal{S}\mathbf{ign}$ (with $\mathbf{\Sigma}_\mathsf{R}$ a subsignature of $\mathbf{\Sigma}_\mathsf{A}$) and every $\mathcal{E}_{Spec}$-sentence ψ on $\mathbf{\Sigma}_\mathsf{A}$ there is an $\mathcal{E}_{Dev}$-sentence $\overline{\vartheta}$ such that:

$$\begin{tikzcd}
\mathbf{\Sigma}_\mathsf{A} \arrow[r, dotted, "i'"] & \mathbf{\Sigma}_\mathsf{C} \\
\mathbf{\Sigma}_\mathsf{R} \arrow[u, "e"] \arrow[r, "i"'] \arrow[ur, phantom, "\mathcal{D}"] & \mathbf{\Sigma}_\mathsf{B} \arrow[u, dotted, "e'"']
\end{tikzcd}$$

(i) $\psi \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_\mathsf{A}} \overline{\vartheta}^e$, and

(ii) for each $\mathcal{E}_{Spec}$-sentence φ on $\mathbf{\Sigma}_\mathsf{B}$, if $\psi^{i'} \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}_\mathsf{C}} \varphi^{e'}$ then $\overline{\vartheta}^i \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_\mathsf{B}} \varphi$.

In addition, $\overline{\vartheta} = \Diamond\!\exists\varkappa_1^{n_1}..\Diamond\!\exists\varkappa_m^{n_m}\psi[p_1 \mapsto \varkappa_1^{n_1}...p_m \mapsto \varkappa_m^{n_n}]$, where $\{p_1,..,p_n\}$ is the difference between $\mathbf{\Sigma}_{\mathsf{A}}$ and $e(\mathbf{\Sigma}_{\mathsf{R}})$.

Moreover, an $\mathcal{E}_{Spec}$-conservative extension $e{:}\langle\mathbf{\Sigma}_{\mathsf{R}},\mathbf{R}\rangle{\to}\langle\mathbf{\Sigma}_{\mathsf{A}},\mathbf{A}\rangle$ are preserved under pushout along an $\mathcal{E}_{Spec}$-interpretation $i{:}\langle\mathbf{\Sigma}_{\mathsf{R}},\mathbf{R}\rangle{\to}\langle\mathbf{\Sigma}_{\mathsf{B}},\mathbf{B}\rangle$, *exactly* when $\mathbf{B}{\vdash}^{\mathcal{E}}_{\mathbf{\Sigma}_{\mathsf{B}}}\mathbf{I}^i(\mathbf{A},\mathbf{B},\mathcal{D})$.

$$\begin{tikzcd}
\langle\mathbf{\Sigma}_{\mathsf{A}},\mathbf{A}\rangle \arrow[r, dotted, "i'"] & \langle\mathbf{\Sigma}_{\mathsf{C}},\mathbf{A}^{i'}\cup\mathbf{B}^{e'}\rangle \\
\langle\mathbf{\Sigma}_{\mathsf{R}},\mathbf{R}\rangle \arrow[u, "e"] \arrow[r, "i"'] \arrow[ur, phantom, "\mathcal{D}"] & \langle\mathbf{\Sigma}_{\mathsf{B}},\mathbf{B}\rangle \arrow[u, dotted, "e'"']
\end{tikzcd}$$

Consequently, if $\mathsf{A} = \langle\mathbf{\Sigma}_{\mathsf{A}},\mathbf{A}\rangle$ is an $\mathcal{E}_{Spec}$-specification, $\mathbf{\Sigma}_{\mathsf{R}}$ is a subsignature of $\mathbf{\Sigma}_{\mathsf{A}}$, and $\varphi_{\mathbf{A}}$ is the conjunction of all the sentences in $\mathbf{A}$ (for recall that specifications are finite theory presentations), then the $\mathcal{E}_{Dev}$-sentence

$$\Diamond\!\exists\varkappa_1^{n_1}...\Diamond\!\exists\varkappa_k^{n_k}\varphi_{\mathbf{A}}[\varsigma_1 \mapsto \varkappa_1^{n_1},...,\varsigma_k \mapsto \varkappa_k^{n_k}]$$

where $\{\varsigma_1,...,\varsigma_k\}$ are the set of $\mathbf{\Sigma}_{\mathsf{A}}$ symbols that appear in the assertion $\varphi_{\mathbf{A}}$ and *do not* belong in $\mathbf{\Sigma}_{\mathsf{R}}$, is an $\mathcal{E}_{Dev}$-axiomatisation of the ($\mathcal{E}_{Spec}$-) $\mathbf{\Sigma}_{\mathsf{R}}$-module of A. In addition, any $\mathcal{E}_{Spec}$-specification $\mathsf{R} = \langle\mathbf{\Sigma}_{\mathsf{R}},\mathbf{R}\rangle$ is classifiable as a $\mathbf{\Sigma}_{\mathsf{R}}$-module of A if $\mathbf{R}{\vdash}^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_{\mathsf{R}}}\Diamond\!\exists\,\varkappa_1^{n_1}...\Diamond\!\exists\,\varkappa_k^{n_k}\varphi_{\mathbf{A}}[\varsigma_1 \mapsto \varkappa_1^{n_1},...,\varsigma_k \mapsto \varkappa_k^{n_k}]$ Hence, by abstracting the "hidden" operators within a development workspace, one can obtain a tractable $\mathbf{\Sigma}_{\mathsf{R}}$-axiomatisation of the $\mathbf{\Sigma}_{\mathsf{R}}$-module for any $\mathcal{E}_{Spec}$-specification A. This axiomatisation can then be used in module design in precisely the same way as an $\mathcal{E}_{Spec}$-axiomatisation of the module would have been used, if it was available. It is just that the proof obligations that assist in controlling the design processes are stated via $\vdash^{\mathcal{E}_{Dev}}$ rather than $\vdash^{\mathcal{E}_{Spec}}$.

Let us examine how the examples of section 3.2 are manipulated within a *Development Workspace*:

If $\mathcal{E}_{Spec}$ is (classical) propositional logic, then the $\mathcal{E}_{Dev}$-presentation of the uniform interpolant, via abstraction of q, for $p \wedge q$ is $\Diamond\!\exists\ \varkappa^0(p \wedge \varkappa^0)$, which is reducible to p, $(p \wedge q) \vee r$ is $\Diamond\!\exists\varkappa^0(p \wedge \varkappa^0) \vee r)$, which is reducible to $p \vee r$, and $p \wedge (q \vee r)$ is $\Diamond\!\exists\varkappa^0(p \wedge (\varkappa^0 \vee r))$, which is reducible to p.

If $\mathcal{E}_{Spec}$ is (classical) first order logic with equality then the $\mathcal{E}_{Dev}$-presentation of the uniform interpolant, via abstraction of f, for $\forall(x)\,(I\,(x) \to O(x, f(x)))$, is $\Diamond\!\exists\varkappa^1(\forall x)(I(x) \to O(x,\varkappa^1(x))\ )$, which is reducible to $(\forall x)(\exists z)(I(x) \to O(x,z))$. Whereas, the $\mathcal{E}_{Dev}$-uniform interpolant for $\forall(x,y)\,(I(x,y) \to O\,(x, f(x), y, g\,(y)))$, via abstraction of f and g, is $\Diamond\!\exists\varkappa_i^1\Diamond\!\exists\varkappa_j^1(\forall x)(\forall y)(I(x) \to O(x,\varkappa_i^1(x), y, \varkappa_j^1(x))\ )$, which is *not* reducible to any first order sentence.

Similarly, for the dense linear order specification one needs to abstract the strict order symbol by an $\Diamond\!\exists$bound 2-ary predicate variable, from the conjunction of the axioms

Uniform schemata, on the other hand, are encapsulated by means of $\boxed{\forall}$ -bound "development" variables: A $\mathbf{\Sigma}_{\mathsf{R}}$-assertion $\boxed{\forall}\,\varkappa_1^{n_1}...\boxed{\forall}\,\varkappa_k^{n_k}\varphi$, where $\boxed{\forall}$ and $\Diamond\!\exists$ do not occur in φ, forces an $\mathcal{E}_{Spec}$-instance $\varphi[\varkappa_1^{n_1} \mapsto \varsigma_1,...,\varkappa_k^{n_k} \mapsto \varsigma_k]$ for any $\varsigma_1...\varsigma_k$ (of the same type as $\varkappa_1^{n_1}...\varkappa_k^{n_k}$) in each $\mathbf{\Sigma}_{\mathsf{A}}$ that includes $\mathbf{\Sigma}_{\mathsf{R}}$. Moreover, a $\mathbf{\Sigma}_{\mathsf{R}}$ sentence ψ of $\mathcal{E}_{Spec}$ is an $\mathcal{E}_{Dev}$-consequence of $\boxed{\forall}\varkappa_1^{n_1}\ldots\boxed{\forall}\varkappa_k^{n_k}\varphi$, whenever $\varphi[\varkappa_1^{n_1} \mapsto \varsigma_1,...,\varkappa_k^{n_k} \mapsto \varsigma_k] \vdash^{\mathcal{E}_{Spec}}_{\mathbf{\Sigma}_{\mathsf{A}}} \psi$ for arbitrary $\mathbf{\Sigma}_{\mathsf{A}}$ that includes $\mathbf{\Sigma}_{\mathsf{R}}$ and arbitrary $\varsigma_1...\varsigma_k$ in $\mathbf{\Sigma}_{\mathsf{A}}$ (of the same type with $\varkappa_1^{n_1}...\varkappa_k^{n_k}$). Hence, for every $\mathbf{\Sigma}_{\mathsf{A}}$ that includes $\mathbf{\Sigma}_{\mathsf{R}}$, we have

$$\boxed{\forall}\varkappa_1^{n_1}\ldots\boxed{\forall}\varkappa_k^{n_k}\varphi \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_{\mathsf{A}}} \psi \quad \textbf{\textit{iff}} \text{ in } \mathbf{\Sigma}_{\mathsf{R}} \text{ we have } \boxed{\forall}\varkappa_1^{n_1}\ldots\boxed{\forall}\varkappa_k^{n_k}\varphi \vdash^{\mathcal{E}_{Dev}}_{\mathbf{\Sigma}_{\mathsf{R}}} \psi$$

ie., the encapsulated schema is uniform.

Revisiting, the main example of section 4, if $\mathcal{E}_{Spec}$ is (classical) predicate logic, then the following $\mathcal{E}_{Dev}$-sentence is a precise encapsulation of the first order induction (uniform) schema:

$$\boxed{\forall}\varkappa^1 \left(\varkappa^1(0) \wedge \left(\forall x\ \varkappa^1(x) \to \varkappa^1(succ(x))\right) \to \forall x\ \varkappa^1(x)\right)$$

Although syntactically similar, this is *not* logically equivalent to, say, second order induction. In particular it is strictly *weaker* than (standard) second order induction. In fact, some aspects of the $\mathcal{E}_{Dev}$-consequence over (classical) predicate logic that have been analysed in [13] reveal that $\mathcal{E}_{Dev}$ is strictly stronger than the infinitary $\mathbf{L}_{\omega_1\omega}$ logic–which is a traditional extension of first order logic–and it is strictly weaker than second order predicate logic–which is another traditional extension of first order logic. This is reflected by the fact that the direction of the following implications is not invertible:

$$\textbf{(U)}\ \vdash^2_{\mathbf{\Sigma}} \forall^2 X^n \mathbf{A}[X^n] \ \Rightarrow\ \vdash^{dev}_{\mathbf{\Sigma}} \boxed{\forall}\varkappa^n \mathbf{A}[\varkappa^n] \ \Rightarrow\ \vdash^{\mathbf{L}_{\omega_1\omega}}_{\mathbf{\Sigma}} \bigwedge\!\!\!\bigwedge \left(\mathbf{A}\left[\varkappa^n \mapsto \chi_i\right]\right)_{i\in\mathbb{F}(n)}$$
$$\textbf{(E)}\ \vdash^{\mathbf{L}_{\omega_1\omega}}_{\mathbf{\Sigma}} \bigvee\!\!\!\bigvee \left(\mathbf{A}\left[\varkappa^n \mapsto \chi_i\right]\right)_{i\in\mathbb{F}(n)} \ \Rightarrow\ \vdash^{dev}_{\mathbf{\Sigma}} \Diamond\!\exists\varkappa^n \mathbf{A}[\varkappa^n] \ \Rightarrow\ \vdash^2_{\mathbf{\Sigma}} \exists^2 X^n \mathbf{A}[X^n]$$

6 Conclusion

We revisited the emergence of a strong connection between module representation and interpolation,and highlighted an analogous connection between module manipulation and uniform schemata. Furthermore, the presence of *uniform interpolants* provide direct and tractable axiomatisations of modules within the accessible language and gives rise to certain proof obligations that assist in establishing the adequacy of the module representation and in verifying the correctness of the *"is module of"* relation. Analogously, the ability to encapsulate and directly manipulate *uniform schemata* supports reasoning with properties of the possibly hidden data via syntactical abstractions. It gives rise to proof obligations that assist in predicting or detecting enrichment via sharing or interaction of "hidden" information, based on the accessible part only. We also remarked that many logics computing *lack* the desirable interpolation properties, and treat (uniform) schematic reasoning as exogenous. To compensate for these inadequacies, we seek for methods to *expand* a specification formalism *orthogonally*, so that an adequately strong version of the critical interpolation properties and a logical encapsulation of (uniform) schemata are obtained.

A potential breakthrough emerges from the framework of a *Development Work-space* described in [13]. The latter is a *Subentailment System* $\langle\mathcal{E}_{Spec}, \mathcal{J}\rangle$ such that $\mathcal{J}{:}\mathcal{E}_{Spec} \to \mathcal{E}_{Dev}$ expands the $\mathcal{E}_{Spec}$-syntax with additional *logical* operators, leaves the extra-logical symbols intact, and extends $\vdash^{\mathcal{E}_{Spec}}$ conservatively. The expansion is *orthogonal* and, not only does $\mathcal{E}_{Dev}$ provides uniform interpolants in derivations between $\mathcal{E}_{Spec}$-sentences, but also, all these crucial interpolants are instances (in $\mathcal{E}_{Dev}$) of a common syntactic form. The availability of a common syntactic form for all the crucial interpolants trivialises the usually complex process of deriving interpolants. The latter is consistent with the insight of [35] and the remark of [8] that uniform interpolants have the same implicit (meta)form. It should be noted that, in our case, this common form is revealed in the formal syntax. On the other hand, the precise encapsulation of uniform schemata by means of abstract operators, provides the essential logical infrastructure for reasoning with "universal" properties of data that *could* be hidden, ie., properties or derivations that are valid in any context that includes the accessible data as a module.

Potential practical applications in Computer Science include the joint extension of systems development environments [45] and algorithmic design tools [41, 44, 43] so that modularity is improved and more advanced and general refinements are supported. Reasoning by means of ($\mathcal{E}_{Dev}$-encapsulations of) uniform schemata may also assist in specifying and developing formal presentations of algorithmic structure via algorithm theories [44]. In fact similar methods are potentially applicable for the presentation and development of several other design abstractions including problem-class specifications and software architecture specifications eg., [48] and [42].

References

1. J. Bergstra adn J. Tucker. Cgaracterization of computable data types by means of a finite equational specification method. In Raj Reddy, editor, *Automata, Languages and Programming, 7^{th} Collogquium*, volume 81 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 1980.
2. G. Tăkeuti. *Proof Theory.* North-Holland Publishing Co., Amsterdam, 1976.
3. H. Andréka, J.F.A.K van Benthem, and I. Németi. Back and Forth between Modal Logic and Classical Logic. *Bulletin of the Interest Group in Pure and Applied Logics*, 1994.
4. David Basin and Seán Matthews. Adding metatheoretic facilities to first-order theories. *Journal of Logic and Computation*, 6(6), 1996.
5. J.A. Bergstra, J. Heering, and P. Klint. Module algebra. *ACM*, 37(2):335–372, 1990.
6. Ritu Chadha. *Applications of Unskolemization.* PhD thesis, University of North Carolina at Chapel Hill, 1991.
7. W. Craig. Three uses of the Herbrand-Getzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic XXII*, pages 269–285, 1957.
8. Giovanna D'Agostino and Marco Hollenberg. Uniform interpolation, automata and the modal μ-calculus, May 1996. Available at http://www.phil.ruu.nl/home/marco/.
9. Răzvan Diaconnescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge University Press, 1993.
10. T. Dimitrakos. On an algebraic flavoring of the logical approach. In *Advances in Theory & Formal Methods of Computing.* ICPress. Distributed by World Scientific Publishing Co., 1996.

11. T. Dimitrakos. An introduction to the *Development Workspace* framework. Technical report, Imperial College, March (revisited September) 1997. Available at http://theory.doc.ic.ac.uk/∼td/publications/.
12. T. Dimitrakos. Craig-Robinson interpolation Modularisation and the construction of refinements. In S. Jähnichen, J. Loeckx, D. Smith, and M. Wirsing, editors, *Dagstuhl Seminar report No. 9710*. TMR-Euroconferences, 1997. Abstract.
13. T. Dimitrakos. *Formal support for specification design and implementation.* PhD thesis, Imperial College, 1997. In preparation.
14. T. Dimitrakos and T.S.E. Maibaum. Notes on refinement, interpolation and uniformity. In *Automated Software Engineering- ASE'97, 12th IEEE International Conference*, 1997. to appear.
15. T. Dimitrakos and T.S.E. Maibaum. On the role of interpolation in stepwise refinement. In *Proceedings of the First Panhellenic Symposium on Logic*, July 1997.
16. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specifications.* EATCS, Monographs on Theoretical Computer Science. Springer-Verlag, 1985.
17. H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.
18. J.L. Fiadeiro and A.Sernadas. Structuring theories on consequence. In D. Sannella and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, pages 44–72, 1988.
19. J.L. Fiadeiro and T.S.E. Maibaum. Stepwise program development in π-institutions. Technical report, Imperial College, 1990.
20. J.L. Fiadeiro and T.S.E. Maibaum. Generalising interpretations between Theories in the Context of (π-)institutions. In S. Gay G. Burn and M. Ryan, editors, *Theory and Formal Methods*, pages 126–147. Springer-Verlag, 1993.
21. K. Gödel. Some metamathematical restults on completeness and consistency; On formally undecidable propositions of *Principia Mathematica* and related systems I; On completeness and consistency. In Jean van Heijnoort, editor, *From Frege to Gödel (A Source Book in Mathematical Logic)*, pages 592–617. Harvard University Press, 1967. Original papers in german (1930-31). Commented translations from german by Stefan Bauer-Megnelberg and Jean van Heijnoort with the permision of Professor Gödel.
22. J. Goguen and Burstall. Introducing institutions. In E Clarke and D. Kozen, editors, *Logics of Programming Workshop*, LNCS 164, pages 221–256. Springer-Verlag, 1984.
23. Joseph A. Goguen. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.
24. Immanuel Kant. *The critique of pure reason; The critique of practical reason and other ethical treatises; The critique of judgement.* Great books of the Western world. 42 Britannica great books. WilliamBenton: Encyclopedia Britanica: Chicago London, 1952.
25. S. Maehara. Craig's interpolation theorem. *Sugaku*, pages 235–237, 1961. In Japanese.
26. T.S.E. Maibaum. The role of abstraction in program development. *Information processing '86*, 1986.
27. T.S.E. Maibaum and M.R. Sadler. Axiomatising specification theory. In H.J. Kreowski, editor, *Recent Trends in Data Type Specification*, pages 171–177. Springer-Verlag, Berlin, 1985.
28. M. Majster. Limits of the algebraic specification of abstract data types. *SIGPLAN Notices*, pages 37–42, October 1977.

29. M. Marx. Interpolation, modularization and knowledge representation. In C. Bioch and Y.H. Tan, editors, *Proceedings of Dutch Conference on AI (NAIC'95)*, June 1995.
30. Jose Meseguer. General logics. In H.D. Ebbinghaus, editor, *Logic Colloquium'87*, pages 275–329, 1989.
31. J. Mesguer and J. Goguen. Initiality, induction and computability. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge, 1985 1985.
32. S. Mikulás. *Taming Logics.* PhD thesis, ILLC-dissertation series 1995-12, 1995.
33. D. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15:1053–1058, 1971.
34. D. Parnas. Information distribution aspects of design methodology. In *Information Processing'72*, volume 71, pages 339–344, 1972. Proceedings of 1972 IFIP Congress.
35. A. Pitts. On an interpretion of second order quantification in first order intuitionistic propositional logic. *Journal of Symbolic Logic*, 57:33–52, 1992.
36. Vasilis Politis, editor. *"Critique of Pure Reason" by Immanuel Kant.* Everyman, London, 1993. A revised and expanded tranlsation based on previous translation by Meiklejohn.
37. P.H. Rodenburg and R.J. van Galbbeek. An Interpolation Theorem in Equational Logic. Technical Report CS-R8838, Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, 1988.
38. D. Sanella and A. Tarlecki. Building theories in an Arbitrary Institution. *Information and Control 76*, 1988.
39. V. Yu. Shavrukov. Subalgebras of diagonalizable algebras of theories containing arithmetic. Dissertationes Mathematicae, Polska Akademia Nauk, Mathematical Institute, Warszawa, 1993.
40. J.R. Shoenfield. *Mathematical Logic.* Addison-Wesley, Publishing Company, 1967.
41. D. R. Smith. KIDS: A semiautomatic program development system. *IEEE Transactions on Software Engineering*, 16(9):1024–1043, September 1990.
42. D. R. Smith. Classification approach to design. Technical report, Kestrel Institute, November 1993.
43. D. R. Smith. Constructing specification morphisms. *Symbolic Computation*, 15(5-6):571–606, May-June 1993.
44. D. R. Smith and M. R. Lowry. Algorithm theories and design tactics. *Science of Computer programming*, 14(2-3):305–321, September 1990.
45. Yellamraju V. Srinivas and Richard Jullig. SpecWare^{TM}: Formal suport for composing software. In *Mathematics of Program Construction*, July 1995. (KES.U.94.5).
46. A. Tarlecki. Bits and pieces of the theory of institutions. In *Workshop on Category Theory and Computer Science*, LNCS 240. Springer-Verlag, 1986.
47. Władysław M. Turski and Thomas S. E. Maibaum. *The Specification of Computer Programs.* Addison-Wesley, 1987.
48. P.A.S. Veloso. Problem solving by interpetation of theories. *Contemporary Mathematics*, 69:241–250, 1988. American Mathematical Society, Providence, Rhode Island.
49. P.A.S. Veloso. Yet another cautionary note on conservative extensions: a simple example with a computing flavour. *Bull. EATCS*, 48:188–192, 1992.
50. P.A.S. Veloso. A new, simpler proof of the modularisation theorem for logical specifications. *Bulletin of the IGPL*, 1(1):3–12, July 1993.

51. P.A.S. Veloso. From extensions to interpretations: Pushout consistency, modularity and interpolation. Technical report, Departamento de inforamtica Pontificia Universidade Catolica do Rio de Janeiro, 1996. (To appear in Information Processing Letters 1997).
52. P.A.S. Veloso and T.S.E. Maibaum. On the modularisation theorem for logical specifications. *Information Processing Letters 53*, pages 287–293, 1995.
53. P.A.S. Veloso, T.S.E. Maibaum, and M.R. Sadler. Program development as theory manipulations. In 3^{rd} *International Workshop on Software Specification and Design*, 1985.
54. P.A.S. Veloso and S. Veloso. On extensions by function symbols: coservativeness and comparison. Technical report, COPPE , Univ. Federal do Rio de Janeiro, July 1990.
55. A. Visser. Bisimulations, model descriptions and propositional quantifiers. Technical report, Department of Philosophy, Utrecht University, 1996.