Synthese (2008) 164:437–450
DOI 10.1007/s11229-008-9358-y

Interpolation in computing science: the semantics of modularization

Gerard R. Renardel de Lavalette

Received: 12 March 2008 / Accepted: 12 June 2008 / Published online: 1 July 2008

Abstract The Interpolation Theorem, first formulated and proved by W. Craig fifty years ago for predicate logic, has been extended to many other logical frameworks and is being applied in several areas of computer science. We give a short overview, and focus on the theory of software systems and modules. An algebra of theories TA is presented, with a nonstandard interpretation of the existential quantifier $\exists$. In TA, the interpolation property of the underlying logic corresponds with the quantifier combination property $\exists\Sigma\ \exists\Pi\ S \equiv \exists(\Sigma \cup \Pi)\ S$. It is shown how the Modularization Theorem, the Factorization Lemma and the Normal Form Theorem for module expressions can be proved in TA.

Keywords Interpolation · Modularization Theorem · Module algebra · Theory algebra

1 Introduction

The applications of logic in computing science are manifold, ranging from electronic circuit design to software design and automated reasoning: see Halpern et al. (2001) for a survey. In this section, we shortly present the applications of the Interpolation Theorem (Craig (1957)) in several areas of computing science, focusing on the logical semantics of specification modules. In the rest of the paper, we develop the theory algebra TA for a unified treatment of the applications of interpolation in this area.

Dedicated to the 50th anniversary of William Craig's Interpolation Theorem.

G. R. Renardel de Lavalette (✉)
Department of Computing Science, University of Groningen, Groningen, The Netherlands
e-mail: g.r.renardel.de.lavalette@rug.nl

1.1 Applications of interpolation: a short overview

We first look at Automated Reasoning, a fruitful area for applications of the interpolation property. Generally speaking, the goal of automated reasoning is: given a collection of formulas S and a formula A in the context of some logic $\mathcal{L}$, determine mechanically whether A follows from S. One way to attack this problem is to generate consequences of S until we obtain A. This may be computationally expensive, especially when S is large. When $\mathcal{L}$ satisfies the interpolation property, a divide-and-conquer strategy can be applied. To explain this, suppose that $S = S_1 \cup S_2$. The idea is to generate consequences B of S_1 which we add to S_2, while at the same time we generate consequences of the extension of S_2 thus obtained until A pops up. Thanks to the interpolation property, we may restrict ourselves to B with $\mathsf{voc}(B) \subseteq \mathsf{voc}(S_1) \cap (\mathsf{voc}(S_2) \cup \mathsf{voc}(A))$. This idea of splitting S can be repeated, yielding a tree with nodes containing sets S_i and edges labelled by signatures that restrict the flow of formulae between adjacent sets. This idea has been worked out in Amir (2002), MacCartney et al. (2003).

In Schlobach (2003, 2004), socalled optimal interpolation is used in the context of terminology construction with description logics. The idea is to find, given concepts C and D subsuming C, a concept I with $\models C \sqsubseteq I \sqsubseteq D$ that can serve as an explanation why D subsumes C. The interpolant I is optimal if not only $\mathsf{voc}(I) \subseteq \mathsf{voc}(C) \cap \mathsf{voc}(D)$, but when $\mathsf{voc}(I)$ is minimal, i.e. there is no I' with $\models C \sqsubseteq I' \sqsubseteq D$ and $\mathsf{voc}(I') \subset \mathsf{voc}(I)$.

In McMillan (2003, 2005, 2006), interpolation is used to speed up symbolic model checking. The goal is to check mechanically whether some temporal logic formula holds in some finite-state model. For this purpose invariants are used to prove properties about inductively defined objects; interpolation is used to filter out information that is not relevant to proving the desired properties.

We turn to the theory of software systems. The realization of a software system can be seen as the transition from a specification of the system (an abstract description of its functionality) to an implementation (a computer program). One approach for obtaining this transition is the method of *stepwise refinement* (see Wirth (1971)), which we explain shortly.

We assume that the specification S_0 is given as a collection of sentences in a logical language $\mathcal{L}$. The idea is that a refinement step brings us from a more abstract description S_0 to a more concrete description S_1 via an interpretation κ of S_0 into a conservative extension T_1 of S_1. More precisely: a *refinement* of S_0 is a specification S_1 with the following properties.

1. S_1 has a *conservative* extension T_1, i.e. if $S_1 \vdash A$ then $T_1 \vdash A$, and if $T_1 \vdash B$ with B in the language of S_1, then $S_1 \vdash B$. Notation: $S_1 \leqslant T_1$.
2. There is an *interpretation* $\kappa : S_0 \to T_1$, i.e. a structure-preserving mapping (so e.g. $\kappa(A \to B) = \kappa(A) \to \kappa(B)$) with $T_1 \vdash A$ for all $A \in S_0$. Notation: $S_0 \stackrel{\kappa}{\to} T_1$.

So we have $S_0 \stackrel{\kappa}{\to} T_1 \geqslant S_1$, and we call this a refinement step.

We sketch an example (based on Maibaum et al. (1985)). Let S_0 be a specification of finite sets of natural numbers, with operations like union, intersection, cardinality, etc.

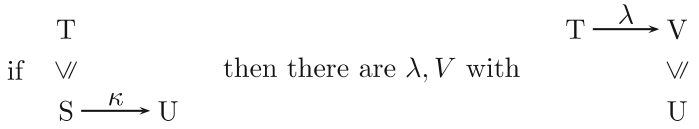

Fig. 1 Diagram of the Modularization Theorem

Sets are not directly representable in computers, but we may consider sets as an abstraction of sequences, and hence refine the notion of sets to the notion of sequences. So let S_1 be a specification of finite sequences of natural numbers, with operations like concatenation of two sequences, taking the first element of a list, etc. The idea is now to extend S_1 to T_1 by adding definable predicates and functions, e.g. the unary predicate `ordered-sequence`, and the binary function `merge` that combines two ordered sequences into their ordered union. Such an extension with definable predicates and functions is evidently conservative. Now it is possible to interpret S_0 in T_1: sets are interpreted by sequences satisfying `ordered-sequence`, union of sets is interpreted by `merge`, etc.

The idea of the method of stepwise refinement is to obtain, via a number of refinement steps

$$S_0 \overset{\kappa_0}{\to} T_1 \geqslant S_1 \overset{\kappa_1}{\to} T_2 \geqslant S_2 \cdots S_{n-1} \overset{\kappa_{n-1}}{\to} T_n \geqslant S_n,$$

a specification S_n that is directly implementable (e.g. a logic program). A natural question arises: is S_n a refinement of S_0? Here the Modularization Theorem enters the stage. It says that (under some mild conditions on S, T, U and κ) if T is a conservative extension of S and $U \vdash \kappa(S)$, then there is a conservative extension V of U and a λ that interprets T in V (see Fig. 1).

From the Modularization Theorem, the transitivity of $\leqslant$ and the composability of interpretations, it follows that a sequence of refinement steps can be reduced to a single refinement step. In Theorem 1, we give a precise formulation and a proof of the Modularization Theorem in Theory Algebra, using the Interpolation property of the underlying logic. Our proof follows the lines of Veloso (1993). A general modularization theorem involving category theory is presented in Dimitrakos and Maibaum (2000).

Another important concept in the development of software systems is the notion of a module. When systems and specifications become large, it is useful and often even necessary to split them into smaller components called modules, which can be developed independently. In the implementation of a module, one often uses items to deal with specific implementation aspects (e.g. depending on hardware and software platform issues), and these items should not be accessible by other modules. This is accomplished by *information hiding* (also called encapsulation): the regulation of the interaction between modules, usually realized by associating to each module an interface that indicates which parts of the module are accessible by the rest of the system.

Logically speaking, information hiding is a kind of existential quantification: if we have a logical specification S containing an auxiliary function f that we want

to hide, then we want to specify '*there is* an f such that S', i.e. $\exists f\ S$. When we consider both information hiding and module composition, the question is how these operations interact. Under what conditions on signature Σ and do they permute, i.e. do we have $\exists\Sigma\ (S \cup T) = (\exists\Sigma\ S) \cup (\exists\Sigma\ T)$? This kind of questions is studied in Module Algebra (MA), presented in Bergstra et al. (1990), which contains the operations $S+T$ (combination of modules S and T) and $\Sigma \,\square\, S$ (exporting signature Σ from S). Export is dual to hiding: $\Sigma \,\square\, S$ yields a module with interface Σ, so in other words all signature elements in S that do not occur in Σ are hidden. In MA, a Factorization Lemma and a Normal Form Theorem are proved. The first says that every extension can be factorized in a conservative extension and a strengthening: if $T \vdash S$ then there is a U with the same signature as S such that T conservatively extends U and $U \vdash S$. The Normal Form Theorem states that every module expression is equivalent to an expression with at most one occurrence of the export operator $\square$. For more details see Theorems 2 and 3 below.

Several semantics for Module Algebra are considered in Bergstra et al. (1990), the most interesting being the theory semantics, where theories embody the meaning of modules. The Theory Algebra TA presented in the next section is inspired on the theory semantics of MA, extended with interpretations. In TA, we derive the Modularization Theorem, the Factorization Lemma and the Normal Form Theorem. We claim that TA is definitely more perspicuous that MA, which has a rather extensive axiomatization. TA and MA are compared in 3.1.

2 Theory algebra

2.1 Preliminaries

We assume some predicate logical language $\mathcal{L}$ to be given over some collection $\mathsf{VOC} = \mathsf{PRED} \cup \mathsf{FUNC}$ of predicate and function symbols (individual constants are identified with function symbols with arity 0). We use A, B, C for formulae in $\mathcal{L}$, and S, T for theories, i.e. subsets of $\mathcal{L}$. For predicate and function symbols in VOC we use p, q and f, g, respectively. Subsets of VOC are called *vocabularies* and denoted by Σ, Π. $\mathsf{voc}(A)$, the vocabulary of A, is the collection of vocabulary elements occurring in A, and $\mathsf{voc}(S) = \bigcup\{\mathsf{voc}(A) \mid A \in S\}$.

The provability relation $S \vdash A$ holds between theories and formulae as usual. We write $S \vdash T$ for ($S \vdash A$ for all $A \in T$) and $\vdash A$ for $\emptyset \vdash A$. Two theories are *equivalent*, notation $S \equiv T$, if $S \vdash T$ and $T \vdash S$. Provability is reflexive ($S \vdash A$ if $A \in S$), transitive (if $S \vdash T$ and $T \vdash A$, then $S \vdash A$), and hence monotonic (if $S \vdash A$ and $S \subseteq T$, then $T \vdash A$).

2.2 Hiding

Theory Algebra TA is a algebraic theory about logical theories (i.e. subsets of $\mathcal{L}$) with three operations: union, hiding, and application of interpretations. We first focus on hiding, postponing the treatment of interpretations. Hiding signature Σ from theory S, denoted by $\exists\Sigma\ S$, is defined by

$$\exists\Sigma\ S = \{A \mid S \vdash A \ \&\ \mathsf{voc}(A) \cap \Sigma = \emptyset\}$$

so $\exists\Sigma\ S$ is the collection of consequences of S that do not contain a vocabulary element in Σ, and we have $\mathsf{voc}(\exists\Sigma\ S) = \mathsf{VOC} - \Sigma$. The notation for hiding is chosen to suggest that it behaves like existential (second order) quantification over vocabulary elements, and we shall show that this is indeed the case. From this perspective, the reader might expect restriction of A to the vocabulary of S in the definition of $\exists\Sigma\ S$, so that $\mathsf{voc}(\exists\Sigma\ S) = \mathsf{voc}(S) - \Sigma$ would hold. That would be in line with our treatment in Renardel de Lavalette (1991), but the present definition is more general, and we shall show that it leads to a rather simple axiomatization where the role of the deduction property and the interpolation property are clearly visible.

The axioms involving hiding are

Mon	if $\Sigma \subseteq \Pi$ and $S \vdash T$ then $\exists\Sigma\ S \vdash \exists\Pi\ T$
Vac	if $\mathsf{voc}(S) \cap \Sigma = \emptyset$ then $S \equiv \exists\Sigma\ S$
Distr	if $\mathsf{voc}(S) \cap \Sigma = \emptyset$ then $\exists\Sigma\ (S \cup T) \equiv S \cup \exists\Sigma\ T$
Comb	$\exists\Sigma\ \exists\Pi\ S \equiv \exists(\Sigma \cup \Pi)\ S$

Mon states the monotonicity of $\exists$ in both arguments: it implies that $\exists$ respects equivalence, i.e. $S \equiv T \Rightarrow \exists\Sigma\ S \equiv \exists\Sigma\ T$. Both Mon and Vac (vacuous quantification) follow directly from the definition of $\exists$ and the monotonicity of $\vdash$. For Distr (conditional distribution) and Comb (quantifier combination), more is required. We shall show that Distr corresponds to the *deduction property*:

if $S \cup T \vdash A$, then there is a $U \subseteq \mathcal{L}$ with
$S \vdash U$, $U \cup T \vdash A$ and $\mathsf{voc}(U) \subseteq \mathsf{voc}(T) \cup \mathsf{voc}(A)$

and Comb to the *interpolation property*:

if $S \vdash A$, then there is a $T \subseteq \mathcal{L}$ with
$S \vdash T$, $T \vdash A$ and $\mathsf{voc}(T) \subseteq \mathsf{voc}(S) \cap \mathsf{voc}(A)$

We observe in passing that, by the compactness of $\vdash$, the deduction property given above follows from ordinary deduction (if $\{A, B\} \vdash C$ then $A \vdash (B \to C)$). To see this, assume $S \cup T \vdash A$ and let $\{B_0, \ldots, B_n\} \subseteq T$ satisfy $S \cup \{B_0, \ldots, B_n\} \vdash A$, then $U := \{(B_0 \wedge \ldots \wedge B_n) \to A\}$ satisfies the required properties.

Lemma 1 (Characterization of deduction) Distr *is equivalent with the deduction property.*

Proof First we prove Distr from the deduction property. Assume $\Sigma \cap \mathsf{voc}(S) = \emptyset$. Now $\exists\Sigma\ (S \cup T) \vdash S \cup \exists\Sigma\ T$ follows from Mon and Vac, so we look at the other direction $S \cup \exists\Sigma\ T \vdash \exists\Sigma\ (S \cup T)$. We must show: if $A \in \exists\Sigma\ (S \cup T)$ then $S \cup \exists\Sigma\ T \vdash A$. So assume $A \in \exists\Sigma\ (S \cup T)$, i.e. $S \cup T \vdash A$ and $\mathsf{voc}(A) \subseteq \mathsf{VOC} - \Sigma$. By the deduction property, we have U with $T \vdash U$, $S \cup U \vdash A$ and $\mathsf{voc}(U) \subseteq \mathsf{voc}(S) \cup \mathsf{voc}(A) \subseteq \mathsf{voc}(S) \cup (\mathsf{VOC} - \Sigma) = \mathsf{VOC} - \Sigma$, (using $\Sigma \cap \mathsf{voc}(S) = \emptyset$ in the last step). It follows that $U \subseteq \exists\Sigma\ T$, so indeed $S \cup \exists\Sigma\ T \vdash A$.

Now the other way round. Assume $S \cup T \vdash A$ and define $\Sigma := \mathsf{VOC} - (\mathsf{voc}(T) \cup \mathsf{voc}(A))$, $U := \exists\Sigma\ S$. Then $\mathsf{voc}(U) = \mathsf{voc}(T) \cup \mathsf{voc}(A)$ and $S \vdash U$, for

$S \equiv \exists\emptyset\ S \vdash \exists\Sigma\ S = U$. Also $U \cup T \vdash A$, for $U \cup T = \exists\Sigma\ S \cup T \equiv \exists\Sigma\ (S \cup T) \vdash \exists\Sigma\ A \equiv A$. This ends the proof.

Lemma 2 (Characterization of interpolation) Comb *is equivalent with the interpolation property.*

Proof We begin with proving Comb. First we observe that $\exists\Sigma\ \exists\Pi\ S \vdash \exists\Sigma\ \exists(\Sigma \cup \Pi)\ S \equiv \exists(S \cup \Pi)\ S$ follows from Mon and Vac. For the other direction, $\exists(S \cup \Pi)\ S \vdash \exists\Sigma\ \exists\Pi\ S$, we argue as follows. Let $A \in \exists\Sigma\ \exists\Pi\ S$, i.e. $\exists\Pi\ S \vdash A$ and $\mathsf{voc}(A) \subseteq \mathsf{VOC} - \Sigma$. By the interpolation property, we find a T with $\exists\Pi\ S \vdash T$, $T \vdash A$ and $\mathsf{voc}(T) \subseteq (\mathsf{VOC} - \Pi) \cap (\mathsf{VOC} - \Sigma) = \mathsf{VOC} - (\Pi \cup \Sigma)$. Hence $T \subseteq \{B \mid S \vdash B\ \&\ \mathsf{voc}(B) \subseteq \mathsf{VOC} - (\Sigma \cup \Pi)\} = \exists(\Sigma \cup \Pi)\ S$ and we get $\exists(\Sigma \cup \Pi)\ S \vdash A$.

Now we prove interpolation from Comb. Assume that $S \vdash A$ and define $\Sigma := \mathsf{VOC} - \mathsf{voc}(S)$, $\Pi := \mathsf{VOC} - \mathsf{voc}(A)$, $T := \exists(\Sigma \cup \Pi)\ S$. Then $\mathsf{voc}(T) = \mathsf{voc}(S) \cap \mathsf{voc}(A)$ and $S \vdash T$, for $S \equiv \exists\emptyset\ S \vdash \exists(\Sigma \cup \Pi)\ S = T$. To see that $T \vdash A$, we argue as follows: $T = \exists(\Sigma \cup \Pi)\ S \equiv \exists\Pi\ \exists\Sigma\ S \equiv \exists\Pi\ S \vdash \exists\Pi\ A \equiv A$ (using Vac in the third step), which ends the proof.

2.3 Interpretations

As mentioned in 1.1, an interpretation is a structure-preserving mapping κ on terms and formulae, i.e. κ commutes with the logical connectives and with the construction of terms and atomic formulae. So e.g. $\kappa(A \wedge B) = \kappa(A) \wedge \kappa(B)$, $\kappa(f(t_1, t_2)) = (\kappa(f))(\kappa(t_1), \kappa(t_2))$, $\kappa(p(t)) = (\kappa(p))(\kappa(t))$. An interpretation κ is fully characterized by an underlying mapping (which we also call κ) on vocabulary elements, that maps k-ary function symbols f to $\lambda x_1, \ldots, x_k.t$ (t some term) and n-ary predicate symbols P to $\lambda x_1, \ldots, x_n.A$ (A some formula).

We extend interpretation application to theories and signatures by

$$\kappa(S) = \{\kappa(A) \mid A \in S\}$$
$$\kappa(\Sigma) = \cup\{\mathsf{voc}(\kappa(s)) \mid s \in \Sigma\}$$

It follows that voc and κ commute, i.e. $\mathsf{voc}(\kappa(S)) = \kappa(\mathsf{voc}(S))$. We define the *domain* and *range* of an interpretation by

$$\mathsf{dom}(\kappa) = \{s \in \mathsf{VOC} \mid s \neq \kappa(s)\}$$
$$\mathsf{rg}(\kappa) = \{\mathsf{voc}(\kappa(s)) \mid s \in \mathsf{dom}(\kappa)\}$$

This is a nonstandard but useful definition of domain, viz. the collection of vocabulary elements that are modified by interpretation κ. For technical reasons, we adopt the following restriction on interpretations:

all interpretations considered satisfy $\mathsf{dom}(\kappa) \cap \mathsf{rg}(\kappa) = \emptyset$

As a consequence, we have $\mathsf{dom}(\kappa) \cap \mathsf{voc}(\kappa(S)) = \emptyset$ for all S. Observe that this restriction excludes interpretations with an inverse (except the identity).

A *renaming* is a special kind of interpretation: its underlying mapping sends function symbols to function symbols, and predicate symbols to predicate symbols, and it is injective on its domain. A renaming ρ has a *pseudo-inverse* ρ^{-1}, satisfying

$\mathsf{dom}(\rho^{-1}) = \mathsf{rg}(\rho)$,
$\mathsf{rg}(\rho^{-1}) = \mathsf{dom}(\rho)$,
$\rho^{-1}(\rho(S)) = S$ for S with $\mathsf{voc}(S) \cap \mathsf{rg}(\rho) = \emptyset$, and
$\rho(\rho^{-1}(T)) = T$ for T with $\mathsf{voc}(T) \cap \mathsf{dom}(\rho) = \emptyset$.

The *equalizer* $E(\kappa)$ of interpretation κ is a theory which expresses that f equals $\kappa(f)$, and p is equivalent to $\kappa(p)$, for all function symbols f and predicate symbols p in the domain of κ:

$$\begin{aligned} E(\kappa) = \quad & \{\forall \vec{x}(p(\vec{x}) \leftrightarrow \kappa(p)(\vec{x})) \mid p \in \mathsf{dom}(\kappa) \cap \mathsf{PRED}\} \cup \\ & \{\forall \vec{x}(f(\vec{x}) = \kappa(f)(\vec{x})) \mid f \in \mathsf{dom}(\kappa) \cap \mathsf{FUNC}\} \end{aligned}$$

We observe that $\mathsf{voc}(E(\kappa)) = \mathsf{dom}(\kappa) \cup \mathsf{rg}(\kappa)$. The following axioms characterize interpretations and their equalizers:

MonInt	if $S \vdash T$ then $\kappa(S) \vdash \kappa(T)$
EqP	$S \cup E(\kappa) \equiv \kappa(S) \cup E(\kappa)$
EqM	$\vdash \kappa(E(\kappa))$

MonInt (monotonicity of interpretations) is a well-known property of predicate logic (and most other logics): observe that it implies that interpretations preserve equivalence, i.e. $S \equiv T \Rightarrow \kappa(S) \equiv \kappa(T)$. EqP is the fundamental property of the equalizer; it follows from $E(\kappa) \vdash t = \kappa(t) \wedge (A \leftrightarrow \kappa(A))$, which is proved straightforwardly with induction over t and A. EqM implies that $E(\kappa)$ is minimal, i.e. it is (modulo $\equiv$) the weakest theory satisfying EqP. For if $S \cup E' \equiv \kappa(S) \cup E'$ for all theories S, then $E(\kappa) \cup E' \equiv \kappa(E(\kappa)) \cup E'$ in particular, so by EqM $E' \vdash E(\kappa)$. For EqM, the restriction to interpretations with $\mathsf{dom}(\kappa) \cap \mathsf{rg}(\kappa) = \emptyset$ is essential: if e.g. $\kappa(p) = \neg p$ then $E(\kappa) \equiv \bot$.

2.4 Properties of TA

Before we turn to the theorems, we formulate and prove some useful properties.

$$S \vdash \exists \Sigma\ S \tag{1}$$

$$\exists \Sigma\ \exists \Pi\ S \equiv \exists \Pi\ \exists \Sigma\ S \tag{2}$$

$$\exists \Sigma\ S \equiv \exists (\Sigma \cap \mathsf{voc}(S))\ S \tag{3}$$

$$\text{if } \Sigma \cap \mathsf{voc}(S) \cap \mathsf{voc}(T) = \emptyset \text{ then } \exists \Sigma\ (S \cup T) \equiv \exists \Sigma\ S \cup \exists \Sigma\ T \tag{4}$$

Property (1) follows from $S \equiv \exists\emptyset\ S \vdash \exists\Sigma\ S$, a direct consequence of Vac and Mon; (2), quantifier shift, is a direct consequence of Comb and the commutativity of $\cup$; and (3) is proved using $\Sigma = (\Sigma \cap \mathsf{voc}(S)) \cup (\Sigma - \mathsf{voc}(S))$, Comb and Vac.

For (4), the conditional distribution of $\exists$ over $\cup$, we reason as follows:

$$
\begin{array}{ll}
 & \exists\Sigma\ (S \cup T) \\
\equiv & \quad \{\Sigma = \Sigma \cup (\Sigma \cap \mathsf{voc}(T)) \text{ and } \mathsf{Comb}\} \\
 & \exists\Sigma\ \exists(\Sigma \cap \mathsf{voc}(T))\ (S \cup T) \\
\equiv & \quad \{\mathsf{Distr}, \text{ using that } \Sigma \cap \mathsf{voc}(S) \cap \mathsf{voc}(T) = \emptyset \text{ is given}\} \\
 & \exists\Sigma\ (S \cup \exists(\Sigma \cap \mathsf{voc}(T))\ T) \\
\equiv & \quad \{(3)\} \\
 & \exists\Sigma\ (S \cup \exists\Sigma\ T) \\
\equiv & \quad \{\mathsf{Distr}, \text{ using } \Sigma \cap \mathsf{voc}(\exists\Sigma\ T) = \Sigma \cap (\mathsf{VOC} - \Sigma) = \emptyset\} \\
 & \exists\Sigma\ S \cup \exists\Sigma\ T
\end{array}
$$

We continue with some properties of interpretations κ and renamings ρ:

$$\vdash \exists\mathsf{dom}(\kappa)\ E(\kappa) \tag{5}$$

$$\kappa(S) \vdash \exists\mathsf{dom}(\kappa)\ S \tag{6}$$

$$\text{if } \mathsf{voc}(S) \cap \mathsf{rg}(\rho) = \emptyset \text{ then } \exists\mathsf{dom}(\rho)\ S \equiv \exists\mathsf{rg}(\rho)\ \rho(S) \tag{7}$$

$$\text{if } \Sigma \cap (\mathsf{dom}(\kappa) \cup \mathsf{rg}(\kappa)) = \emptyset \text{ then } \kappa(\exists\Sigma\ S) \equiv \exists\Sigma\ \kappa(S) \tag{8}$$

To see that (5) holds, observe that (1) implies $E(\kappa) \vdash \exists\mathsf{dom}(\kappa)\ E(\kappa)$, so with MonInt we have $\kappa(E(\kappa)) \vdash \kappa(\exists\mathsf{dom}(\kappa)\ E(\kappa))$; but $\vdash \kappa(E(\kappa))$ (EqM) and $\kappa(\exists\mathsf{dom}(\kappa)\ E(\kappa)) = \exists\mathsf{dom}(\kappa)\ E(\kappa)$, and we conclude $\vdash \exists\mathsf{dom}(\kappa)\ E(\kappa)$.

Equation (6) can be read as the *instantiation property* of $\exists$, when we interpret $\kappa(S)$ as a substitution instance of S with respect to the function and predicate symbols in $\mathsf{dom}(\kappa)$. To see that it holds, observe that $\kappa(S) \equiv \kappa(S) \cup \exists\mathsf{dom}(\kappa)\ E(\kappa) \equiv \exists\mathsf{dom}(\kappa)\ (\kappa(S) \cup E(\kappa)) \vdash \exists\mathsf{dom}(\kappa)\ S$, using (5), Distr and EqP, respectively.

Property (7) is proved in two steps. Assume that $\mathsf{voc}(S) \cap \mathsf{rg}(\rho) = \emptyset$, so $\rho^{-1}(\rho(S)) = S$. First we observe

$$
\begin{array}{ll}
 & \exists\mathsf{dom}(\rho)\ S \\
\vdash & \quad \{S = \rho^{-1}(\rho(S)), \text{ and } (6)\} \\
 & \exists\mathsf{dom}(\rho)\ \exists\mathsf{dom}(\rho^{-1})\ \rho(S) \\
\equiv & \quad \{\mathsf{dom}(\rho^{-1}) = \mathsf{rg}(\rho), \text{ and } (2)\} \\
 & \exists\mathsf{rg}(\rho)\ \exists\mathsf{dom}(\rho)\ \rho(S) \\
\equiv & \quad \{\mathsf{Vac} \text{ using } \mathsf{dom}(\rho) \cap \mathsf{voc}(\rho(S)) = \emptyset\} \\
 & \exists\mathsf{rg}(\rho)\ \rho(S)
\end{array}
$$

So we have $\exists\mathsf{dom}(\rho)\ S \vdash \exists\mathsf{rg}(\rho)\ \rho(S)$ for all ρ and S, hence $\exists\mathsf{dom}(\rho^{-1})\ \rho(S) \vdash \exists\mathsf{rg}(\rho^{-1})\ \rho^{-1}(\rho(S))$ in particular: this proves the other direction of (7), using $\mathsf{dom}(\rho^{-1}) = \mathsf{rg}(\rho)$, $\mathsf{rg}(\rho^{-1}) = \mathsf{dom}(\rho)$ and $\rho^{-1}(\rho(S)) = S$.

Finally we prove (8), the conditional permutation of κ and $\exists$:

$$
\begin{array}{ll}
 & \kappa(\exists\Sigma\ S) \\
\equiv & \quad \{(5) \text{ and } \mathsf{Distr}, \text{ using } \mathsf{dom}(\kappa) \cap \mathsf{voc}(\kappa(\exists\Sigma\ S)) = \emptyset\}
\end{array}
$$

$$
\begin{array}{ll}
 & \exists \mathsf{dom}(\kappa)\ (\kappa(\exists\Sigma\ S) \cup E(\kappa)) \\
\equiv & \quad \{\mathsf{EqP}\} \\
 & \exists \mathsf{dom}(\kappa)\ (\exists\Sigma\ S \cup E(\kappa)) \\
\equiv & \quad \{\mathsf{Distr} \text{ using } \Sigma \cap \mathsf{voc}(E(\kappa)) = \Sigma \cap (\mathsf{dom}(\kappa) \cup \mathsf{rg}(\kappa)) = \emptyset, \text{ and (2)}\} \\
 & \exists\Sigma\ \exists \mathsf{dom}(\kappa)\ (S \cup E(\kappa)) \\
\equiv & \quad \{\mathsf{EqP}\} \\
 & \exists\Sigma\ \exists \mathsf{dom}(\kappa)\ (\kappa(S) \cup E(\kappa)) \\
\equiv & \quad \{\mathsf{Distr}, \text{ using } \mathsf{dom}(\kappa) \cap \mathsf{voc}(\kappa(S)) = \emptyset, \text{ and (5)}\} \\
 & \exists\Sigma\ \kappa(S)
\end{array}
$$

We introduce the *export operator* $\boxdot$ as an abbreviation:

$$\Sigma \boxdot S = \exists(\mathsf{VOC} - \Sigma)\ S$$

So $\Sigma \boxdot S = \{A \mid S \vdash A \ \&\ \mathsf{voc}(A) \subseteq \Sigma\}$, the collection of consequences of S with vocabulary in Σ; also $\mathsf{voc}(\Sigma \boxdot S) = \Sigma$. We have the following useful property for switching between $\exists$ and $\boxdot$:

$$\text{if } \mathsf{voc}(S) - \Sigma = \mathsf{voc}(S) \cap \Pi \text{ then } \Sigma \boxdot S \equiv \exists\Pi\ S \tag{9}$$

which follows from (3) and the definition of $\boxdot$. With $\boxdot$, we can reformulate the definition of $\geqslant$ (conservative extension):

$$T \geqslant S \quad \text{iff } S \equiv \mathsf{voc}(S) \boxdot T$$

which we use in the proofs of the theorems in the next subsection.

2.5 Theorems

Theorem 1 (Modularization) *Let theories S, T, U and interpretation κ be given with* $\mathsf{voc}(S) \subseteq \mathsf{voc}(T)$, $\mathsf{voc}(T) \cap \mathsf{voc}(U) = \emptyset$, $\mathsf{dom}(\kappa) = \mathsf{voc}(S)$ *and* $\mathsf{rg}(\kappa) \subseteq \mathsf{voc}(U)$. *Then the following pushout property holds:*

$$\textit{if } T \geqslant S,\ S \xrightarrow{\kappa} U, \textit{ then there is a minimal } V \textit{ with } T \xrightarrow{\kappa} V,\ V \geqslant U$$

Proof Assume $T \geqslant S$ and $S \xrightarrow{\kappa} U$, i.e. $\mathsf{voc}(S) \boxdot T \equiv S$ and $U \vdash \kappa(S)$. We shall show that $V := U \cup \kappa(T)$ satisfies $T \xrightarrow{\kappa} V$ and $V \geqslant U$, and that it is the least V doing so. $V \vdash \kappa(T)$ is evident, so $T \xrightarrow{\kappa} V$. For $V \geqslant U$, i.e. $\mathsf{voc}(U) \boxdot V \equiv U$, we argue as follows, using $\Sigma := \mathsf{voc}(T) - \mathsf{voc}(S)$.

$$
\begin{array}{ll}
 & \mathsf{voc}(U) \boxdot V \\
\equiv & \quad \{\text{definition of } V\} \\
 & \mathsf{voc}(U) \boxdot (U \cup \kappa(T)) \\
\equiv & \quad \{(9) \text{ using } \mathsf{voc}(U \cup \kappa(T)) - \mathsf{voc}(U) = \Sigma = \mathsf{voc}(U \cup \kappa(T)) \cap \Sigma\} \\
 & \exists\Sigma\ (U \cup \kappa(T)) \\
\equiv & \quad \{\mathsf{Distr}, \text{ using } \Sigma \cap \mathsf{voc}(U) = \emptyset\}
\end{array}
$$

$$
\begin{array}{ll}
 & U \cup (\exists \Sigma\ \kappa(T)) \\
\equiv & \quad \{(8), \text{using } \Sigma \cap (\mathsf{dom}(\kappa) \cup \mathsf{rg}(\kappa)) = \emptyset\} \\
 & U \cup \kappa(\exists \Sigma\ T) \\
\equiv & \quad \{(9), \text{using } \mathsf{voc}(T) - \mathsf{voc}(S) = \Sigma = \mathsf{voc}(T) \cap \Sigma\} \\
 & U \cup \kappa(\mathsf{voc}(S) \boxdot T) \\
\equiv & \quad \{S \leqslant T \text{ is given, i.e. } \mathsf{voc}(S) \boxdot T \equiv S\} \\
 & U \cup \kappa(S) \\
\equiv & \quad \{S \xrightarrow{\kappa} U \text{ is given, i.e. } U \vdash \kappa(S)\} \\
 & U
\end{array}
$$

To show that V is minimal wrt. $\vdash$, assume that $T \xrightarrow{\kappa} V'$ and $V' \geqslant U$, i.e. $V' \vdash \kappa(T)$ and $U \equiv \mathsf{voc}(U) \boxdot V'$; this last equivalence implies $V' \vdash U$, so we have $V' \vdash U \cup \kappa(T)$, i.e. $V' \vdash V$.

The next theorem is called the Factorization Lemma in Bergstra et al. (1990): *every extension can be split in an enrichment and a refinement*. An enrichment is a conservative extension, and a refinement T of S is defined in Bergstra et al. (1990) by: $T \vdash S$ and $\mathsf{voc}(T) = \mathsf{voc}(S)$ (observe that this differs from the notion of refinement that we use in this paper).

Theorem 2 (Factorization lemma) *If $T \vdash S$, then there is a U such that $T \geqslant U$, $U \vdash S$ and $\mathsf{voc}(U) = \mathsf{voc}(S)$.*

Proof Easy: take $U = \mathsf{voc}(S) \boxdot T$ and the result follows, using that $\mathsf{voc}(U) = \mathsf{voc}(\mathsf{voc}(S) \boxdot T) = \mathsf{voc}(S)$.

Theorem 3 (Normal Form) *Any theory expression built from theories and the operators $\cup$, $\exists$ and $\boxdot$, can be rewritten in an equivalent expression containing no $\boxdot$ and at most one occurrence of $\exists$ (or: no $\exists$ and at most one occurrence of $\boxdot$).*

Proof Without loss of generality, we assume that VOC is large enough, i.e. contains, for every k, infinitely many predicate and function symbols with arity k not occurring in the theory expression under consideration.

The idea is: first replace all occurrences of $\boxdot$ by $\exists$ using $\Sigma \boxdot S \equiv \exists(\mathsf{voc}(S) - \Sigma)\ S$, then push all occurrences of $\exists$ outwards using the combination property

$$\exists \Sigma\ S \cup \exists \Pi\ T \equiv \exists(\Sigma \cup \Pi)\ (S \cup T) \tag{10}$$

and finally combine all occurrences of $\exists$ using $\exists \Sigma\ \exists \Pi\ S \equiv \exists(\Sigma \cup \Pi)\ S$ (i.e. Comb) to obtain one occurrence of $\exists$ (which may be converted to $\boxdot$ via $\exists \Sigma\ S \equiv (\mathsf{VOC} - \Sigma) \boxdot S$).

To realize this idea, we have to prove (10). However, it turns out that (10) only holds under the condition that $\mathsf{voc}(S) \cap \Pi = \mathsf{voc}(T) \cap \Sigma = \emptyset$, which does not hold in general. We shall use renamings to enforce the required disjointness, but first we prove (10) under the condition $\mathsf{voc}(S) \cap \Pi = \mathsf{voc}(T) \cap \Sigma = \emptyset$:

$$
\begin{array}{ll}
 & \exists(\Sigma \cup \Pi)\ (S \cup T) \\
\equiv & \quad \{(4) \text{ using } (\Sigma \cup \Pi) \cap \mathsf{voc}(S) \cap \mathsf{voc}(T) = \emptyset\}
\end{array}
$$

$$
\begin{array}{ll}
 & \exists(\Sigma \cup \Pi)\ S \cup \exists(\Sigma \cup \Pi)\ T \\
\equiv & \quad \{\mathsf{Comb}\ \} \\
 & \exists\Sigma\ \exists\Pi\ S \cup \exists\Pi\ \exists\Sigma\ T \\
\equiv & \quad \{\mathsf{Vac} \text{ using } \mathsf{voc}(S) \cap \Pi = \mathsf{voc}(T) \cap \Sigma = \emptyset\} \\
 & \exists\Sigma\ S \cup \exists\Pi\ T
\end{array}
$$

We shall show how to enforce the condition $\mathsf{voc}(S) \cap \Pi = \mathsf{voc}(T) \cap \Sigma = \emptyset$ with help of renamings. Let S, T, Σ, Π be given. Now take σ, π with $\mathsf{dom}(\sigma) = \Sigma$, $\mathsf{dom}(\pi) = \Pi$, $(\mathsf{rg}(\sigma) \cup \mathsf{rg}(\pi)) \cap (\mathsf{voc}(S) \cup \mathsf{voc}(T)) = \mathsf{rg}(\sigma) \cap \mathsf{rg}(\pi) = \emptyset$: such σ, π can always be found, thanks to the assumption at the beginning of the proof. Then $\mathsf{rg}(\sigma) \cap \mathsf{voc}(\pi(T)) = \mathsf{rg}(\pi) \cap \mathsf{voc}(\sigma(S)) = \emptyset$ and

$$
\begin{array}{ll}
 & \exists\Sigma\ S \cup \exists\Pi\ T \\
\equiv & \quad \{\mathsf{dom}(\sigma) = \Sigma, \mathsf{dom}(\pi) = \Pi\} \\
 & \exists\mathsf{dom}(\sigma)\ S \cup \exists\mathsf{dom}(\pi)\ T \\
\equiv & \quad \{(7), \text{using that } \mathsf{voc}(S) \cap \mathsf{rg}(\sigma) = \mathsf{voc}(T) \cap \mathsf{rg}(\pi) = \emptyset\} \\
 & \exists\mathsf{rg}(\sigma)\ \sigma(S) \cup \exists\mathsf{rg}(\pi)\ \pi(T) \\
\equiv & \quad \{(10) \text{ using } \mathsf{rg}(\sigma) \cap \mathsf{voc}(\pi(T)) = \mathsf{rg}(\pi) \cap \mathsf{voc}(\sigma(S)) = \emptyset\} \\
 & \exists(\mathsf{rg}(\sigma) \cup \mathsf{rg}(\pi))\ (\sigma(S) \cup \pi(T))
\end{array}
$$

This ends the proof of the Normal Form Theorem.

3 Final remarks

3.1 Comparison with module algebra

As was explained in Sect. 1, TA is inspired on Module Algebra (MA) with the theory semantics as defined in Bergstra et al. (1990). An important difference between TA and MA, which is responsible for the relatively simple axiomatization of TA, is the nature of the semantic objects. In MA with the theory semantics, these are formula sets $S \subseteq \mathcal{L}$ that are $\vdash$-closed with respect to their vocabulary, i.e. they satisfy

$$S = \{A \mid S \vdash A, \mathsf{voc}(A) \subseteq \mathsf{voc}(S)\} \tag{11}$$

and the equality relation is literal identity. In TA, however, the semantic objects are *arbitrary* theories $S \subseteq \mathcal{L}$, with provable equivalence $\equiv$ as equality relation.

The operators of MA are $S+T$ (combination of modules S and T), $\Sigma \,\square\, S$ (export of a signature Σ from module S), $\mathsf{T}(\Sigma)$ (the minimal module with signature Σ) and $r.S$ (renaming module S with r). Only involutive renamings r are allowed, i.e. satisfying $r(r(S)) = S$. In the theory semantics, the operators are defined as follows:

$$
\begin{aligned}
S + T &= \{A \mid S \cup T \vdash A \ \&\ \mathsf{voc}(A) \subseteq \mathsf{voc}(S) \cup \mathsf{voc}(T)\} \\
\Sigma \,\square\, S &= \{A \mid S \vdash A \ \&\ \mathsf{voc}(A) \subseteq \Sigma \cap \mathsf{voc}(S)\} \\
\mathsf{T}(\Sigma) &= \{A \mid \vdash A \ \&\ \mathsf{voc}(A) \subseteq \Sigma\} \\
r.S &= \{r(A) \mid A \in S\}
\end{aligned}
$$

It is easily verified that the operators of MA preserve $\vdash$-closure with respect to the vocabulary, i.e. property (11). Here the restriction to involutive renamings is essential.

The operators of MA are definable in TA:

$$S + T = (\mathsf{voc}(S) \cup \mathsf{voc}(T)) \boxdot (S \cup T)$$
$$\Sigma \square S = (\Sigma \cap \mathsf{voc}(S)) \boxdot S$$
$$\mathsf{T}(S) = \mathsf{voc}(S) \boxdot \emptyset$$

Moreover, the combination and export operators of MA and TA appear to be equivalent: $S + T \equiv S \cup T$ and $\Sigma \square S \equiv \Sigma \boxdot S$. The first equivalence follows from Mon and Vac, so without using deduction or interpolation of the underlying logic. The second equivalence $\Sigma \square S \equiv \Sigma \boxdot S$, however, requires Comb and hence interpolation. As a consequence, the theorems formulated in 2.5 also hold when we read $+$ for $\cup$ and $\square$ for $\boxdot$.

The axioms of MA involving interpolation are

$E3$ $\quad \Sigma \square (\mathsf{T}(\Pi) + S) = \mathsf{T}(\Sigma \cap \Pi) + \Sigma \square S$

$E4$ $\quad$ if $\mathsf{voc}(S) \cap \mathsf{voc}(T) \subseteq \Sigma$ then $\Sigma \square (S + T) = \Sigma \square S + \Sigma \square T$

E3 follows from the interpolation property, but E4 requires the following *strong interpolation property*:

if $S \cup T \vdash A$, then there is a $U \subseteq \mathcal{L}$ with
$S \vdash U$, $U \cup T \vdash A$ and $\mathsf{voc}(U) \subseteq \mathsf{voc}(S) \cap (\mathsf{voc}(T) \cup \mathsf{voc}(A))$

This property is equivalent to the conjunction of the deduction property and the ordinary interpolation property that we formulated above. There is, however, no axiom in MA that corresponds to the deduction property only. In TA, E3 follows from $\Sigma \square S \equiv \Sigma \boxdot S$ and $\Sigma \boxdot \emptyset \equiv \emptyset$, and E4 is equivalent to (4).

3.2 Uniform interpolation

In TA, the following *uniform* interpolation property holds:

for every S and $\Sigma \subseteq \mathsf{voc}(S)$, there is a theory U with
$S \vdash U$, $\mathsf{voc}(U) \subseteq \Sigma$ and
for all T: if $S \vdash T$ and $\mathsf{voc}(S) \cap \mathsf{voc}(T) \subseteq \Sigma$, then $U \vdash T$.

This is easily proved with $U := \Sigma \boxdot S$, using Mon, Vac and Comb. So ordinary interpolation in the underlying logic yields uniform interpolation in TA. On the other hand, uniform unterpolation in the underlying logic is equivalent with the property *hiding preserves finiteness*: if S is finite, then there is is a finite T with $T \equiv \exists \Sigma\ S$.

3.3 Alternatives for predicate logic

We developed TA with predicate logic as the underlying logic, but we only used the reflexivity and transitivity of $\vdash$, the preservation of $\vdash$ under interpretations, and the deduction and the interpolation property. This means that any other logic with these

properties can be used as underlying logic. But if we e.g. replace predicate logic by equational logic, we must give up Comb, for equational logic has the interpolation property (see Rodenburg (1991, 1992), Renardel de Lavalette (2005)) but not the deduction property. We give a counterexample: it is evident that

$$f(a) = b,\ f(c) = d, a = c \vdash b = d$$

so according to the deduction property there should be a collection U of equations with $f(a) = b,\ f(c) = d \vdash U, U \cup \{a = c\} \vdash b = d$ and $\mathsf{voc}(U) \subseteq \{a, b, c, d\}$. But such a U does not exist in the language of equational logic.

3.4 Conclusion

We presented TA, an algebraic theory about logical theories with the operators union, hiding and application of interpretations. TA has seven axioms: five follow from natural properties of the derivability relation $\vdash$, the two other axioms correspond exactly with the deduction property and the interpolation property of the underlying logic, respectively. The operator $\exists$ (hiding) satisfies several properties of existential quantification: quantifier shift $\exists\Sigma\ \exists\Pi\ S \equiv \exists\Pi\ \exists\Sigma\ S$, vacuous quantification $\mathsf{voc}(S) \cap \Sigma = \emptyset \Rightarrow \exists\Sigma\ S \equiv S$, existential instantiation $\kappa(S) \vdash \exists\mathsf{dom}(\kappa)\ S$ and conditional distribution $\mathsf{voc}(S) \cap \Sigma = \emptyset \Rightarrow \exists\Sigma\ (S \cup T) \equiv S \cup \exists\Sigma\ T$. TA captures several important properties that are relevant for the logical semantics of specification modules, among them the Modularization Theorem.

References

Amir, E. (2002). *Dividing and conquering logic*. PhD thesis. Stanford University.

Bergstra, J. A., Heering, J., & Klint, P. (1990). Module algebra. *Journal of the ACM, 37*, 335–372.

Craig, W. (1957) Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic, 22*, 250–268.

Dimitrakos, T., & Maibaum, T. (2000). On a generalized Modularization Theorem. *Information Processing Letters, 74*, 65–71.

Halpern, J. Y., Harper, R., Immerman, N., Kolaitis, P. G., Vardi, M. Y., & Vianu, V. (2001). On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic, 7*, 213–236.

MacCartney, B., McIlraith, S. A., Amir, E., & Uribe, T. (2003). Practical partition-based theorem proving for large knowledge bases. In G. Gottlob & T. Walsh (Eds.), *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)* (pp. 89–98). Morgan Kaufmann.

Maibaum, T., Veloso, P., & Sadler, M. (1985). A theory of abstract data types for program development: Bridging the gap? In H. Ehrig, C. Floyd, M. Nivat, & J. W. Thatcher (Eds.), *Mathematical Foundations of Software Development, Proceedings of TAPSOFT, Berlin, Volume 2: Colloquium on Software Engineering (CSE)*, volume 186 of *Lecture Notes in Computer Science* (pp. 214–230). Springer-Verlag.

McMillan, K. L. (2003). Interpolation and SAT-based model checking. In W. A. Hunt Jr. & F. Somenzi (Eds.), *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8–12, 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science* (pp. 1–13). Springer-Verlag.

McMillan, K. L. (2005). An interpolating theorem prover. *Theoretical Computer Science, 345*(1), 101–121.

McMillan, K. L. (2006). Lazy abstraction with interpolants. In T. Ball & R. B. Jones (Eds.), *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17–20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science* (pp. 123–136). Springer-Verlag.

Renardel de Lavalette, G. R. (1991). Logical semantics of modularization. In E. Börger, G. Jäger, H. Kleine Büning, & M. M. Richter (Eds.), *Computer Science Logic '91*, volume 626 of *Lecture Notes in Computer Science* (pp. 306–315). Springer-Verlag.

Renardel de Lavalette, G. R. (2005). Abstract derivations, equational logic and interpolation (extended abstract). In P. Bruscoli, F. Lamarche, & C. Stewart (Eds.), *Structures and deduction—the quest for the essence of proofs (satellite workshop of ICALP 2005)* (pp. 173–188). Technische Universität Dresden, Fakultät Informatik, 2005. Technical Report FI05-08-Juli 2005, ISSN 1430-211X.

Rodenburg, P. (1991). A simple algebraic proof of the equational interpolation theorem. *Algebra Universalis, 28*, 48–51.

Rodenburg, P. (1992). Interpolation in equational logic. Technical report, University of Amsterdam, Department of Mathematics and Computer Science, Programming Research Group, January 1992. Report P9201.

Schlobach, S. (2003). Optimal interpolation in ALC. In *Proceedings of Methods for Modalities 3, Nancy, France*. http://m4m.loria.fr/M4M3/Papers/schlobach.ps.gz.

Schlobach, S. (2004). Explaining subsumption by optimal interpolation. In J. J. Alferes & J. Leite (Eds.), *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)* (pp. 413–425) LNAI 3229.

Veloso, P. A. S. (1993). A new, simpler proof of the modularisation theorem for logical specifications. *Bulletin of the IGPL, 1*, 3–12.

Wirth, N. (1971). Program development by stepwise refinement. *Communications of the ACM, 14*, 221–227.