PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 33/95

Logical Specifications: 4.A - Interpretations of Unsorted Specifications

Paulo A. S. Veloso
Thomas S. E. Maibaum

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 33/95
Editor: Carlos J. P. Lucena October, 1995

Logical Specifications: 4.A - Interpretations of Unsorted Specifications *

Paulo A. S. Veloso
Thomas S. E. Maibaum

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio — Departamento de Informática
Rua Marquês de São Vicente, 225 — Gávea
22453-900 — Rio de Janeiro, RJ
Brasil
Tel. +55-21-529 9386 Telex +55-21-31048 Fax +55-21-511 5645
E-mail: rosane@inf.puc-rio.br

LOGICAL SPECIFICATIONS: 4.A INTERPRETATIONS OF UNSORTED SPECIFICATIONS

Paulo A. S. VELOSO and **Thomas S. E. MAIBAUM**
{e-mail: veloso@inf.puc-rio.br and tsem@doc.ic.ac.uk}

PUCRioInf MCC 33/94

Abstract. We present basic concepts and results concerning interpretations of unsorted logical specifications, as well as some connections with extensions. We start with symbol-to-symbol interpretations, examining syntactic translations of terms and formulae, semantical induction of structures, and interpretations of specifications. We then examine some operations on specifications and interpretations: union and intersection of specifications over orthogonal languages, composition and decompositions of interpretations, a simple internalisation technique via kernel, and the unsorted Modularisation Construction. We next consider the Modularisation Theorem for unsorted specifications, first the case of extensions, and then its extension to interpretations. Finally, we consider some variants of interpretations often encountered in the literature: translations of predicates and operations to formulae, translation of variables, and briefly relativisation and translation of equality. This report is the first part of a draft of the fourth section of a handbook chapter. Other reports cover the remaining sections.

Key words: Formal specifications, unsorted axiomatic specifications, interpretations of specifications, unsorted language translations, internalisation of translation, internalised kernel, conservative extensions, modularity, interpolation, translations of predicates, operations and variables, relativisation, translation of equality.

Resumo. Apresentamos conceitos e resultados básicos sobre interpretações de especificações lógicas sem sortes, bem como algumas conexões com extensões. Começamos com interpretações símbolo-a-símbolo, examinando traduções sintáticas de termos e formulas, indução semântica de estruturas, e interpretações de especificações. A seguir, consideramos algumas operações em especificações e interpretações: união e interseção de especificações sobre linguagens ortogonais, composição e decomposições de interpretações, uma técnica simples de internalização através de núcleo e a versão sem sortes da Construção de Modularização. Tratamos então do Teorema da Modularização para especificações sem sortes, partindo do caso de extensões, que é a seguir estendido a interpretações. Por fim, consideramos algumas variantes de interpretações comuns na literatura: traduções de predicados e operações a formulas, tradução de variáveis e brevemente relativização e tradução da igualdade. Este relatório é a primeira parte de um esboço da quarta seção de um capítulo de um manual. Outros relatórios cobrem as demais seções.

Palavras chave: Especificações formais, especificações axiomáticas sem sortes, interpretações de especificações, traduções de linguagens sem sortes, internalização de tradução, núcleo internalizado, extensões conservativas, modularidade, interpolação, traduções de predicados, operações e variáveis, relativização, tradução da igualdade.

NOTE

This report is the first part of a draft of the fourth section of a chapter in a forthcoming volume of the Handbook of Logic in Computer Science

Other reports, corresponding to the remaining sections, have been issued or are in preparation. The plan of the chapter - and series of reports Logical Specifications - is as follows.

1. Introduction and Overview — MCC 13/94, June 1994, (v+11 p)
2. Specifications as Presentations — MCC 26/94, July 1994, (vi+24)
3. Extensions of Specifications — MCC 33/94, Sept. 1994, (vi+58)
4. Interpretation of Specifications — in two parts: A and B
5. Implementation of Specifications
6. Parameterised Specifications
7. Conclusion: Retrospect and Prospects.

The chapter - and series of reports - is intended to provide an account of the logical approach to formal specification development.

Any comments or criticisms will be greatly appreciated.

The next report in this series is planned to be Logical Specifications: 4.B Interpretations of many-sorted Specifications, covering the many-sorted version of the present one, namely:

many-sorted translations and interpretations;
composition, decomposition and internalisations of translations;
variations of translations and interpretations, relativisation, translation of equality, sort tupling;
joint consistency and modularity of extensions and of interpretations.

ACKNOWLEDGEMENTS

Research reported herein is part of an on-going research project. Partial financial support from British, European Community and Brazilian agencies is gratefully acknowledged. The hospitality and support of the institutions involved have been very helpful. Collaboration with Martin R. Sadler, Sheila R. M. Veloso and José L. Fiadeiro was instrumental in sharpening many ideas. The authors would like to thank the following colleagues for many fruitful discussions on these and related topics: Carlos J. P. de Lucena, Samit Khosla, Atendolfo Pereda Bórquez, Douglas R. Smith, Haydée W. Poubel, M. Claudia Meré, Tarcísio H. C. Pequeno and Roberto Lins de Carvalho

CONTENTS *

* See the preceding note for an explanation of the numbering system

List of Figures

4 Interpretations of Specifications [1]

Specification engineering involves the study and practice of specifications: construction, refinement, implementation, etc. As mentioned in the introduction, an implementation step involves an interpretation (and an extension, see 1.3), as does parameter instantiation (see 1.5). We have already examined extensions in section 3; we shall now consider interpretations, as well as some other aspects of extensions.

Aside from the technical details of the definition, the most important property of interpretations is that they preserve properties, a requirement that is crucial for the use of abstractions. One reasons about a program on the basis of some axiomatisation of the abstract objects used in this program; when these objects are implemented by more concrete objects, one expects that the conclusions reached at the abstract level will still hold. Otherwise one would lose all the work done at the abstract level, presumably having to redo it (or some part of it) at the more concrete level. This is in general much more difficult because of the extra details used at the concrete level and is exactly what one wanted to avoid by the use of abstraction and stepwise refinement.

In implementing abstract objects in terms of more concrete ones, we are generally dealing with distinct languages. For instance, we may choose to represent sets (of natural numbers, say) by lists (of natural numbers). Since these languages may be quite different, we must begin by providing a common ground for comparison; this can be done by means of a translation of one language to the other. Once this is provided, we can compare the specifications with respect to their consequences or their realisations.

Translation is a very important activity in program development; an automatic form of this activity is compiling. The translations underlying interpretations bear some similarity to the binding established between a procedure invocation and its definition.

Because we are interested here in the concept of specification and identify specifications with theory presentations, we wish to explore the translations within the same logical system: first-order logic. A largely neglected, but for computer science very important, area of concern is the study of translations between theories in different logical systems.

This section presents the concepts and results pertaining to translations and interpretations in two stages. We first examine the unsorted case (from 4.2 to 4.5) and then extend this treatment to the many-sorted case

---

[1] See the preceding note for an explanation of the terminology 'chapter', 'section', etc., as well as for the numbering system.

(from 4.6 to 4.9)[1].

The presentation of the unsorted case is as follows. We start in 4.2 with symbol-to-symbol interpretations, examining syntactic translations of terms and formulae, semantical induction of structures, and interpretations of specifications. We then examine in 4.3 some operations on specifications and interpretations: union and intersection of specifications over orthogonal languages, composition and decompositions of interpretations, a simple internalisation technique, and the unsorted Modularisation Construction. In 4.4 we consider the Modularisation Theorem for unsorted specifications, first the case of extensions in 4.4.1, which is extended to interpretations in 4.4.2. Then, we consider in 4.5 some variants of interpretations: translations of predicates and operations to formulae, translation of variables, and briefly relativisation and translation of equality.

The presentation of the many-sorted version follows basically the same outline as the unsorted case. We begin in 4.6 with symbol-to-symbol interpretations, examining syntactic translations of terms and formulae, semantical induction of structures, and interpretations of specifications. We then consider in 4.7 some operations on specifications and interpretations. Union and intersection of specifications over orthogonal languages as well as composition and decompositions of interpretations are the expected extensions of their unsorted versions; we also present a more powerful internalisation technique, and then many-sorted Modularisation Construction. In 4.8 we consider the Modularisation Theorem for many-sorted specifications, first some variations in 4.8.1, then modularity of extensions in 4.8.2, which is extended to interpretations in 4.8.3. Then, we consider in 4.9 some variants of many-sorted interpretations and their connections with the sort-introducing constructs seen in 3.8. We first examine translations with relativisation (reduced to subsort) in 4.9.1, translation of equality (reduced to quotient) in 4.9.2, translation with sort tupling (reduced to product) in 4.9.3. We then consider general interpretations, combining the above three variants, in 4.9.4, and finally examine in 4.9.6 the reduction of many-sorted logic to unsorted logic.

Some examples in 4.10 illustrate these ideas.

4.1 Introduction

Several somewhat different notions of interpretation have been proposed in the literature (Enderton 1972; Shoenfield 1967; Veloso and Pequeno 1978; Goguen *et al.* 1978; Turski and Maibaum 1987), and it is not quite clear which one is most appropriate. In the case of first-order logic, an

[1] The many-sorted case in presented in the report Logical Specifications: 4.B. Interpretations of many-sorted Specifications.

obvious candidate has been developed by logicians: interpretations between (unsorted) theories (Enderton 1972, p. 156-163; Shoenfield 1967, p. 61-62). Computer scientists have extended this idea to many-sorted first-order logic, and it is this that we will discuss below. Some efforts have also been made to develop appropriate notions of translation for equational logics (Ehrig et al. 1982, Ehrich 1982).

Now, the exact definitions of translation/interpretation present some variation in technical details. The common factors underlying these concepts can be identified, however, as including the following:

1. The logical symbols need no translation as they form the common infrastructure for the theories under consideration. Thus, translations concentrate on describing the extra-logical symbols of one theory in terms of those of another.
2. Translations between theories will induce translations between corresponding models. Some (abstract) objects in a model of the theory being translated may be represented by more that one (concrete) object in a model of the target theory. In our example above, we may choose to represent a set (of natural numbers) by any list (of natural numbers) with no repetition of values in the list, as long as it contains the 'same' natural numbers as the set. So, equality may need translation.
3. Given the induced translation between models, we may have some concrete objects in the model of the target theory which are not used to represent any objects in the model of the abstract theory. We will need to differentiate between representatives and such non representatives within models of the target theory in order to make sense of property preservation. The concept of relativisation predicate is introduced to cope with this problem.
4. Translation of formulae involving quantifiers may be tricky in the following sense. Quantifiers usually relate the quantified variable to the range of possible values in structures. Since formulae are used to express properties, when these formulae are translated, care must be taken to make sure that the translated formulae are restricted in the domain of values to which they might potentially be applied. The concept of relativisation is again pertinent.

We now proceed in stages to define for first-order logic the concepts of language translation and specification interpretation. We shall first deal with the unsorted case, and its variations, and then proceed to the many-sorted case.

4.2 Unsorted symbol interpretations

We shall now examine unsorted language translations and specification

interpretations. We first concentrate on the simple case of symbol to symbol translation, and later consider some possible variations.

We begin by considering translations of languages, which are mappings enabling syntactical translations of properties expressed in the language (Shoenfield 1967). We then examine how these translations also induce mappings between the corresponding structures - albeit in the 'reverse' direction - and establish a Translation Connection: a formula and its translation "express the same thing" (Enderton 1972).

4.2.1 Syntactic translation

We first examine the simple case of a translation between (unsorted) languages - sometimes called interpretation of languages (Shoenfield 1967, p. 61, 62; Turski and Maibaum 1987, p. 82, 265) or signature morphism (Ehrich 1982) - as a mapping between symbols enabling syntactical translations of terms and formulae.

A (symbol) translation from a language L' to language L" is a mapping t that assigns to each symbol of the former a symbol of the latter, respecting their syntaxes. More precisely, a *(symbol) translation* (or *renaming* or *signature morphism*) I from *source language* L' to *target language* L" - denoted $I:L' \rightarrow L"$ - is a mapping $I:Alph(L') \rightarrow Alph(L")$, composed of predicate renaming and operation renaming, both respecting syntactical declarations, in the following sense:

- if r is an m-ary predicate symbol of L' then I(r) is an m-ary predicate symbol of L";
- if f is an n-ary operation symbol of L' then I(f) is an n-ary operation symbol of L".

Note that mapping I is not necessarily surjective nor injective. For the unsorted case, we do not have to translate the variables; but we may. (If we do, the translation of variables is to be injective, as explained below.)

The translation I of symbols of L' to L" can be naturally extended to translate terms and formulae. The *translation of a term* $t \in Trm(L')$ - respectively, *formula* $\varphi \in Frml(L')$ - via I is the term $I(t) \in Trm(L")$ - respectively, formula $I(\varphi) \in Frml(L")$ - obtained by the replacing each occurrence of a symbol of L' by its translation according to the mapping I. More precise definitions of translation of term and formula are formulated by induction (Shoenfield 1967, p. 61, 62).

A special case is obtained when language L' is a sub-language of L"; then the inclusion mapping $J:L' \rightarrow L"$ is a symbol translation.

4.2.2 Semantical translation

We now examine how a translation of languages induces a natural connection between structures (albeit in the 'reverse' direction) by giving

rise to an induced structure (Enderton 1972). The motivation is being able to discuss structures of a language in terms of another language (Shoenfield 1967, p. 61).

A translation from a language to another one permits terms and formulae of the former to be translated to terms and formulae of the latter. We would like to claim that the original formula and its translation "say the same thing". But, they are formulae in different languages. For this claim to make sense, we must relate the structures in these two languages.

When language L' is a sub-language of L", the inclusion mapping J:L'→L" is a symbol translation and the reduct of a structure for L" is a structure for L' (see 3.3). We shall now see how this natural connection generalises to language translations.

Let I:L'→L" be a (symbol) translation from source language L' to target language L". A structure for L" assigns realisations to its symbols; by composing the translation with this assignment, we obtain realisations for the symbols of L' (see figure 4.1). Given a structure $\mathfrak{M}$" for target language L", the *structure induced* by $\mathfrak{M}$" under I - denoted $\mathfrak{M}$"↓I - is the structure $\mathfrak{M}$' for source language L', such that

- its universe M' is the universe of $\mathfrak{M}$": M'=M";
- for each predicate symbol r of L', its realisation in the induced structure $\mathfrak{M}$"↓I is the realisation of its translation I(r) in the given structure $\mathfrak{M}$": $\mathfrak{M}$'[r]=$\mathfrak{M}$"[I(r)];
- for each operation symbol f of L', its realisation in the induced structure $\mathfrak{M}$"↓I is the realisation of its translation I(f) in the given structure $\mathfrak{M}$": $\mathfrak{M}$'[f]=$\mathfrak{M}$"[I(f)].

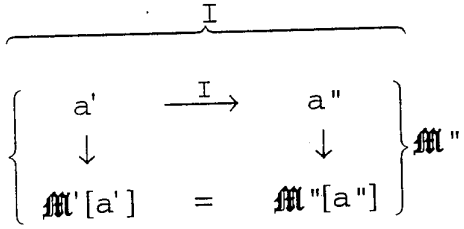

Fig. 4.1: Structure induced by translation

Notice that a translation of a language to another one induces structures in the "reverse" direction: from the latter to the former (Enderton 1972, p. 159). Similarly, notice that any assignment a:Var(L")→M" of values in the domain M" of the given structure $\mathfrak{M}$" to the variables of L" is, in view of the above construction, an assignment of values in the domain M'=M" of the induced structure $\mathfrak{M}$"↓I to the variables of L'.

The clause indicating how the realisation of an operation symbol f of L'

in the induced structure $\mathfrak{M}''\downarrow I$ comes from its realisation in $\mathfrak{M}''$ is $\mathfrak{M}''\downarrow I[f]=\mathfrak{M}''[I(f)]$. This extends inductively to any term $t\in Trm(L')$: $\mathfrak{M}''\downarrow I[t]=\mathfrak{M}''[I(t)]$. Similarly, for any formula $\varphi\in Frml(L')$, $\mathfrak{M}''\downarrow I[\varphi]=\mathfrak{M}''[I(\varphi)]$: the relation defined by the translated formula $I(\varphi)$ on the given structure $\mathfrak{M}''$ and the relation defined by the original formula φ on the induced structure $\mathfrak{M}''\downarrow I$ are the same. This is the - expected - content of the Translation Connection (Enderton 1972, p. 161), which indicates that one can discuss structures for one language in the other.

Proposition Translation Connection (unsorted case)

Let $I:L'\rightarrow L''$ be a (symbol) translation from source language L' to target language L". Consider a structure $\mathfrak{M}$ for L" inducing $\mathfrak{M}\downarrow I$.

Then, for every assignment $a:Var(L'')\rightarrow M$ of values in the universe of M to the variables, we have

a) for each term t of source language L': $\mathfrak{M}\downarrow I[t][a]=\mathfrak{M}[I(t)][a]$;

b) for each formula φ of source language L': $\mathfrak{M}\downarrow I\models\varphi\ [a]$ iff $\mathfrak{M}\models I(\varphi)\ [a]$.

In particular, for every sentence $\sigma\in Sent(L')$: $\mathfrak{M}\downarrow I\models\sigma$ iff $\mathfrak{M}\models I(\sigma)$.

4.2.3 Interpretations of specifications

We shall now consider interpretations of specifications as translations of languages that preserve the properties proved (Shoenfield 1967; Turski and Maibaum 1987). We shall also establish the Interpretation Theorem, which characterises interpretations by means of axioms, theorems and models, and use it to characterise faithfulness.

Consider specifications $P'=<L',G'>$ and $P''=<L'',G''>$. A *symbol interpretation* from *source specification* P' to *target specification* P" - denoted $I:P'\rightarrow P''$ - is a language translation $I:L'\rightarrow L''$ preserving logical consequences, in the following sense: for every sentence σ of L', if $P'\models\sigma$ then $P''\models I(\sigma)$. In other words, I translates the source theory Cn[P'] into the target theory Cn[P"]: $I(Cn[P'])\subseteq Cn[P'']$.

An example of symbol interpretation is provided by extension of specifications. The analogue of conservative extension is a *faithful* interpretation: one where $P'\models\sigma$ iff $P''\models I(\sigma)$ for every sentence σ of L'.

In view of the Translation Connection, a symbol translation $I:L'\rightarrow L''$ maps logically valid formulae of L' to logically valid formulae of L". Thus, we may regard a language translation $I:L'\rightarrow L''$ as a specification interpretation from $Trv(L'):=<L',\varnothing>$ to $Trv(L''):=<L'',\varnothing>$. This is roughly the idea of interpretation of languages (Shoenfield 1967, p. 61).

We now characterise which language translations are interpretations, by means of axioms and (induced) models. This is the content of the next

result, implication (b⇒a) of which is sometimes called the Interpretation Theorem (Shoenfield 1967, p. 62; Turski and Maibaum 1987, p. 85). Assertion (b) gives a useful property-oriented criterion for interpretation, assertion (c) being its model-oriented counterpart.

Theorem Interpretation Theorem (for unsorted specifications)

Consider a language translation I:L'→L". Given specifications P'=<L',G'> and P"=<L",G">, the following are equivalent.

a) I is an interpretation from P' to P".

b) For every axiom $\gamma \in G'$, its translation $I(\gamma)$ is a consequence of P".

c) For every $\mathfrak{M} \in Mod[P"]$, the induced structure $\mathfrak{M}\downarrow I$ is a model of P'.

Proof.

(a⇒b) Clear, because $G' \subseteq Cn[P']$.

(b⇒c) Given a structure $\mathfrak{M} \in Mod[P"]$, we shall show that $\mathfrak{M}\downarrow I \in Mod[P"]$. For this purpose, consider an axiom $\gamma \in G' \subseteq Cn[P']$. By (b), $P" \models I(\gamma)$, whence $\mathfrak{M} \models I(\gamma)$. But then, by the Translation Connection, $\mathfrak{M}\downarrow I \models \gamma$.

(c⇒a) Given a theorem $\tau \in Cn[P']$, we will show $P" \models I(\tau)$. For this purpose, consider a model $\mathfrak{M} \in Mod[P"]$,. By (c), $\mathfrak{M}\downarrow I \in Mod[P"]$, thus $\mathfrak{M}\downarrow I \models \tau$. But then, by the Translation Connection, $\mathfrak{M} \models I(\gamma)$. Hence, $\mathfrak{M} \models I(\gamma)$, whenever $\mathfrak{M} \in Mod[P"]$.

QED

As a consequence, if I interprets <L',G'> into <L",G">, then I also interprets <L',G'∪H> into <L",G"∪I(H)>, for any set $H \subseteq Sent(L')$. We can also characterise which interpretations are faithfull.

Proposition Characterisation of faithfulness (for unsorted specifications)

Consider an interpretation from P'=<L',G'> to P"=<L",G">. Then, the following are equivalent.

a) Interpretation I:P'→P" is faithfull.

b) For every $H \subseteq Sent(L')$, I interprets <L',G'∪H> faithfully into <L",G"∪I(H)>.

c) For every $H \subseteq Sent(L')$, if <L',G'∪H> is consistent, then so is <L",G"∪I(H)>.

d) For every structure $\mathfrak{A} \in Mod[P']$, there exists $\mathfrak{B} \in Mod[P']$, such that $\mathfrak{A}$ is elementarily equivalent to the structure induced by $\mathfrak{B}$: $\mathfrak{A} \equiv \mathfrak{B}\downarrow I$.

Proof.

(a⇒b) Consider $\sigma \in Sent(L')$ with $I(\sigma) \in Cn[<L",G"\cup I(H)>]$. Then, by compactness, there exist sentences $\theta_1, \ldots, \theta_k \in H$, with $I(\sigma)$ in

$Cn[<L'',G''\cup\{I(\theta_1),\ldots,I(\theta_k)\}>]$. Letting θ be the conjunction of $\theta_1,\ldots,\theta_k$, we have $H\models\theta$ and, by the Deduction Theorem, $I(\theta\rightarrow\sigma)\in Cn[<L'',G''>]$, with $(\theta\rightarrow\sigma)\in Sent(L')$. Thus, $(\theta\rightarrow\sigma)\in Cn[<L',G'>]$ by (a), whence $\sigma\in Cn[<L',G'\cup H>]$.

(b$\Rightarrow$c) Assume $<L'',G''\cup I(H)>$ is inconsistent and let $\sigma\in Sent(L')$. Then $I(\sigma\wedge\neg\sigma)=I(\sigma)\wedge\neg I(\sigma)\in Cn[<L'',G''\cup I(H)>]$. Thus $\sigma\wedge\neg\sigma\in Cn[<L',G'\cup H>]$ by (b).

(c$\Rightarrow$d) Given $A\in Mod[P']$, $<L',G'\cup Th(\mathfrak{A})>$ is a consistent extension of P'. Thus, by (c) $<L'',G''\cup I[Th(\mathfrak{A})]>$ is consistent, so has some model $\mathfrak{B}$. Now, $\mathfrak{B}\in Mod[P'']$ and $\mathfrak{B}\downarrow I\models Th(\mathfrak{A})$, so $\mathfrak{A}\equiv\mathfrak{B}\downarrow I$.

(d$\Rightarrow$a) Given $\tau\in Sent(L')$ with $I(\tau)\in Cn[P'']$, we will show $\tau\in Cn[P']$. Given a model $\mathfrak{A}\in Mod[P']$, we have some $\mathfrak{B}\in Mod[P'']$ with $\mathfrak{A}\equiv\mathfrak{B}\downarrow I$ by (d). Thus, $\mathfrak{B}\models I(\tau)$, and so $\mathfrak{B}\downarrow I\models\tau$, whence $\mathfrak{A}\models\tau$. Hence, $\mathfrak{A}\models\tau$ whenever $\mathfrak{A}\in Mod[P']$.

QED

4.3. Operations on (unsorted) specifications and interpretations

We shall now examine some concepts and notations which will be useful in the sequel. These have to do with some simple operations on specifications, composition and decompositon of interpretations and a simple internalisation technique. We also introduce the (unsorted) Modularisation Construction.

4.3.1 Operations on (unsorted) specifications

We will now examine some simple operations on languages and specifications, as well as orthogonal families of languages.

A. *Union and intersection (unsorted case)*

We first review some simple ideas concerning languages and specifications within the same context.

Imagine that a symbol of a language L', say a predicate symbol, happened to be a symbol of a different kind in another language L", say an operation symbol. Then, considering the two languages in the same context would lead to a somewhat confusing situation. This would not occur if both L' and L" happened to be sub-languages of some language L; we then call L' and L" *compatible*. This is usually the situation when we wish to form the intersection or union of languages.

Consider an unsorted language L with alphabet A and set of variables V. Given a sub-alphabet $A'\subseteq A$, we can form the sub-language of L with alphabet A' and set of variables V; we will call this sub-language the *restriction of* L *to sub-alphabet* $A'\subseteq A$ and denote it by $L|A'$. Notice that, for each $v\in V$, $v\approx v$ is a formula of $L|A'$.

Now, consider a family of sub-languages $L_i\subseteq L$, for $i\in I$. Then

$Alph(L_i) \subseteq Alph(L)$ and $Var(L_i)=Var(L)$. We then form

- the *intersection language* $\cap_{i\in I} L_i$ as the restriction of L to sub-alphabet $\cap_{i\in I} Alph(L_i) \subseteq Alph(L)$; and
- the *union language* $\cup_{i\in I} L_i$ as the restriction of L to sub-alphabet $\cup_{i\in I} Alph(L_i) \subseteq Alph(L)$.

Notice that the intersection and the union language have the same set of variables as the given (unsorted) languages.

We say that sub-languages L' and L" of L are *disjoint* (which we denote by $L' \cap L" = \emptyset$) when they have no common extra-logical symbol ($Alph(L') \cap Alph(L") = \emptyset$), sharing only variables and the common equality.

Given a family of specifications $P_i = \langle L_i, G_i \rangle$ over sub-languages $L_i \subseteq L$, for $i \in I$, by the *union specification* we mean $\cup_{i\in I} P_i := \langle \cup_{i\in I} L_i, \cup_{i\in I} G_i \rangle$.

B. *Orthogonal families of languages*

We will now examine orthogonal families of languages and their joint expansiveness property.

Orthogonality is a condition for "union without confusion". By an *orthogonal family of languages* we mean a family of sub-languages L_i of some language L, for $i \in I$, such that whenever $j \neq k$ in I we have $L_j \cap L_k \subseteq L_\cap$, where $L_\cap := \cap_{i\in I} L_i$ is the intersection of the family L_i.

Orthogonality is also a condition for joint expandabilitiy of structures for languages of the family to structures for the union language. We shall use the notations $a \in L$ for 'a is a symbol of language L' and $\mathfrak{A} \subseteq \mathfrak{B}$ for 'structure $\mathfrak{B}$ expands $\mathfrak{A}$'.

Lemma Joint expandabilitiy for orthogonal family

Given an orthogonal family of languages $L_i \subseteq L$, for $i \in I$, let $L_\cap := \cap_{i\in I} L_i$ and $L^\cup := \cup_{i\in I} L_i$. Consider a family of structures $\mathfrak{B}_i$ for L_i, for $i \in I$, such that any two $\mathfrak{B}_j$ and $\mathfrak{B}_k$ have a common reduct $\mathfrak{B}_j \downarrow L_\cap = \mathfrak{B}_k \downarrow L_\cap$ to $L_\cap$. Then, there exists a structure $\mathfrak{C}$ for $L^\cup$ that is a common expansion of the given $\mathfrak{B}_i$, for $i \in I$.

Proof.

Given a symbol a of $L^\cup$, let $I(a) := \{i \in I / a \in L_i\}$; so $a \in I(a)$. By orthogonality, for each symbol $a \in L^\cup$ we have: either $a \in L_\cap$ and I(a)=I; or else I(a) is a singleton {i} for some $i \in I$. We have $t: L^\cup \rightarrow I \cup \{*\}$ such that, for $a \in L_\cap$, t(a)=*, and otherwise $I(a)=\{t(a)\}$. Now, let $L_* := L_\cap$ and $\mathfrak{B}_*$ be the common reduct of the B_i to $L_\cap$. We define structure $\mathfrak{C}$ for $L^\cup$ by $\mathfrak{C}[a] := \mathfrak{B}_{t(a)}[a]$, since $a \in L_{t(a)}$. We will then have $\mathfrak{B}_j \subseteq \mathfrak{C}$, for each $i \in I$.

Indeed, given $a \in L_i$, either $a \in L_\cap$ and $\mathfrak{C}[a]=\mathfrak{B}_*[a]=\mathfrak{B}_i[a]$, or $\mathfrak{C}[a]=\mathfrak{B}_{t(a)}[a]=\mathfrak{B}_i[a]$.

QED

Union of an orthogonal family preserves expansiveness, in the sense of the next result.

Proposition Orthogonal expansiveness

Given an orthogonal family of languages L_i, for $i \in I$, let $L_\cap := \cap_{i \in I} L_i$ and $L^\cup := \cup_{i \in I} L_i$. Consider a specification $P=<L_\cap, G>$ and a family $P_i=<L_i, G_i>$ of extensions of P, for $i \in I$. Assume that for every $i \in I$ we have an expansive extension $P \preceq P_i$. Then, the union specification $P^\cup=<L^\cup, \cup_{i \in I} G_i>$ is an expansive extension of P, as well: $P \preceq P^\cup$.

Proof.

A model $\mathfrak{A} \in Mod[P]$ has an expansion $\mathfrak{B}_i \in Mod[P_i]$ for each $k \in I$.

We thus have a family of structures $\mathfrak{B}_i \in Mod[P_i]$ for L_i, $i \in I$, with a common reduct $\mathfrak{A}$ to $L_\cap$. By the preceding lemma, this family $\{\mathfrak{B}_i / i \in I\}$ has a common expansion $\mathfrak{C}$ to $L^\cup$.

To see that $\mathfrak{C} \in Mod[P^\cup]$, consider an axiom $\gamma \in \cup_{i \in I} G_i$.

Then, $\gamma \in G_i$, for some $i \in I$, and $\mathfrak{B}_i \models \gamma$. Hence $\mathfrak{C} \models \gamma$, since $\mathfrak{B}_i \subseteq \mathfrak{C}$.

QED

4.3.2 Composition and decomposition of interpretations

We shall now briefly examine composition and decomposition of translations and of interpretations. We shall consider two kinds of decompositions: some with a more 'global' nature and some of a more 'local' character.

A. *Composition of interpretations*

Composability of (symbol) translations and interpretations is to be expected.

Consider (symbol) translations $I': L \rightarrow L'$ and $I'': L' \rightarrow L''$. Clearly the composite mapping $I';I''$ assigns to each symbol of L a corresponding symbol of L'' of the same kind, being thus a (symbol) translation $I';I'': L \rightarrow L''$ from source language L to target language L''.

Similarly, given (symbol) interpretations $I': P \rightarrow P'$ and $I'': P' \rightarrow P''$, involving specifications $P=<L,G>$, $P'=<L',G'>$ and $P''=<L'',G''>$, we have successive preservation of theorems, which makes the composite $I';I''$ an interpretation from source specification P to target specification P''.

More interesting than composition is decomposing a translation. We shall examine two kinds of decompositions: first those with a more 'global nature', via pre and post images; and then those of a more 'local' character,

induced by partitioning the source language.

B. *Decompositions via pre and post images*

The Interpretation Theorem (for Specifications) suggests an interesting byproduct: the idea of pre and post images of a specification under a translation. (A model-based concept of preimage is used by Enderton (1972, p. 159). These will provide decompositions for interpretations with a more global character, which will be useful in the sequel.

Consider a translation $I:L' \rightarrow L''$. Given a specification $P'=<L',G'>$ over source language L', consider the set $I(G') := \{I(\gamma) \in Sent(L'')/\gamma \in G'\}$ of the translations of its axioms. This gives a specification over the target language, namely $I(P'):=<L'',I(G')>$, called the *postimage* of P' *under translation* I. Also, given a specification $P''=<L'',G''>$ over target language L'', consider the set $I^{-1}(Cn[P'']):=\{\sigma \in Sent(L')/I(\sigma) \in Cn[P'']\}$ of sentences of L' that I translates to theorems of P''. This gives a (not necessarily finite) presentation of a theory, namely $I^{-1}(P''):=<L',I^{-1}(Cn[P''])>$, called the *preimage* of P'' *under translation* I.

The pre and post images of a specification are meant to characterise interpretations. The preimage of a target specification characterises the specifications that the translation can interpret into the given specification (Enderton 1972, p. 162), whereas the postimage of a source specification characterises the specifications into which the translation can interpret the given specification. This is made precise in the following result, whose assertions turn out to be reformulations of the Interpretation Theorem.

Proposition Characterisation of interpretation by pre and post images

Consider a translation $I:L' \rightarrow L''$ from language L' to language L''.

a) Given a specification $P'=<L',G'>$, its postimage $I(P')=<L'',I(G')>$ under I is the weakest specification over L'' into which I interprets P', in that:

(i) I is an interpretation from P' to I(P'), and

(ii) any specification P over L'' that I interprets P' into P is an extension of I(P'): $I(P') \subseteq P$.

b) Given $P''=<L'',G''>$, its preimage $I^{-1}(P'')=<L',I^{-1}(Cn[P''])>$ under I is the strongest specification over L' that I interprets into P'', in that:

(i) I is an interpretation from $I^{-1}(P'')$ to P'', and

(ii) $I^{-1}(P'')$ extends any specification P over L' that I interprets into P'': $P \subseteq I^{-1}(P'')$.

Pre and post images also give decompositions of an interpretation akin to that of a function via its kernel and image, as illustrated in figure 4.2.

Corollary Decomposition of interpretation via pre and post images

Consider an interpretation I from P'=<L',G'> into P"=<L",G">. We then have postimage I(P')=<L",I(G')> and preimage $I^{-1}(P")$=<L',I^{-1}(Cn[P"])>.

a) Interpretation I:P'→P" can be decomposed via postimage into an interpretation I':P'→I(P') followed by an extension e":I(P')⊆P".

b) Interpretation I:P'→P" can be decomposed via preimage into an extension e':P'⊆I^{-1}(P") followed by a faithful interpretation I":I^{-1}(P")→P".

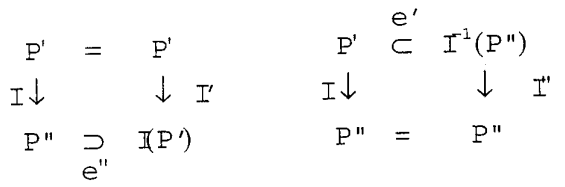

Fig. 4.2: Decomposition of interpretation via pre and post images

C. *Decompositions based on language partitions*

We shall now consider decompositions of a more 'local' character, induced by partitions of the source alphabet. Such decompositions will indicate that one can treat each extra-logical symbol separately, which is convenient, both for theoretical and practical purposes.

Consider a (symbol) translation I from source language L' to target language L", and assume them to be disjoint. Consider a partition Alph(L')=$A^*\cup A^\#$, which induces sub-languages L^*:=L'|A^* and $L^\#$:=L'|$A^\#$ so that L'=$L^*\cup L^\#$. We then have two translations I^* and $I^\#$, as follows (see figure 4.3):

- translation I^* acts as I over L^* and acts identically on the reamining symbols;
- translation $I^\#$ acts as I over $L^\#$ and is the identity on the reamining symbols.

The composite map $I^*;I^\#$ is a translation with the same behaviour as the original I, which has thus been decomposed into two translations, each one acting on a sub-language of L'.

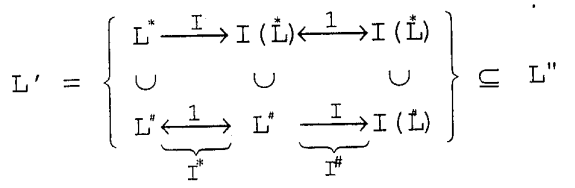

Fig. 4.3: Decomposition of a translation

Now, given specifications P'=<L',G'> and P"=<L",G"> such that I:P'→P" is an interpretation, the above decomposition yields a specification P=I^*[P'],

together with interpretations $I^*:P' \rightarrow P$ and $I^\#:P \rightarrow P''$ decomposing I.

For our present purposes, the main point of the above decompositions is that they permit translating each extra-logical symbol separately, which will simplify some considerations in the sequel.

4.3.3 Internalisation of a translation via the kernel

We shall now introduce an internalisation technique, coding part of the information provided by a translation, which was conceived to prove the Modularisation Theorem. This technique will internalise the kernel of a translation, by coding its information into sentences of the source language. As such, it reduces, to some extent, interpretations to extensions (Veloso 1992; Veloso and Maibaum 1992). Another internalisation technique, via the diagram (Veloso 1993), will be considered when dealing with many-sorted interpretations.

Translations consist of mappings from source to target symbols. As usual, the *kernel* of a mapping $I:Alph(L') \rightarrow Alph(L'')$ is the (equivalence) relation $ker[I]:=\{<a_1,a_2>:I(a_1)=I(a_2)\}$ We shall now show how this concept of kernel can be 'internalised' within the source language (Veloso 1992; Veloso and Maibaum 1992).

We can 'internalise' the kernel by coding its information into the set $\Lambda[I]=\{(a_1 \leftrightarrow a_2):a_1,a_2 \in Alph(L') \& I(a_1)=I(a_2)\}$ of sentences of L' - expressing the equivalence of source symbols with the same translation. More precisely, for each pair $<a_1,a_2>$ of symbols of source language L', we construct its *identifying sentence* $\lambda[I]<a_1,a_2> \in Sent(L')$, as follows:

- for each pair $<r_1,r_2>$ of predicate symbols of L' - with the same arity, say m - its *identifying sentence* $\lambda[I]<r_1,r_2>$ is the sentence of L': $\forall v_1 \ldots \forall v_m[r_1(v_1,\ldots,v_m) \leftrightarrow r_2(v_1,\ldots,v_m)]$;
- for each pair $<f_1,f_2>$ of operation symbols of L' - with the same arity, say n - its *identifying sentence* $\lambda[I]<f_1,f_2>$ is the sentence of L': $\forall x_1 \ldots \forall x_n[f_1(x_1,\ldots,x_n) \approx f_2(x_1,\ldots,x_n)]$.

We now collect the appropriate identifying sentences into the set $\Lambda[I] := \{\lambda[I]<a_1,a_2> \in Sent(L'):<a_1,a_2> \in ker[I]\}$, called the *identifying diagram of translation* $I:L' \rightarrow L''$. By the (*internalised*) *kernel of translation* $I:L' \rightarrow L''$ we mean the specification $Krnl[I]:=<L',\Lambda[I]>$.

For a translation $I:L' \rightarrow L''$, its (internalised) kernel $Krnl[I]=<L',\Lambda[I]>$ provides part of the information given by the mapping, namely which symbols have the same translation, without providing the translations themselves. This information will be seen to be enough to characterise faithfulness.

An injective translation $I:L' \rightarrow L''$ applied to a specification $P'=<L',G'>$ will

just copy the axioms in G' onto I(G'); one thus expects the interpretation $I:P' \to I(P')$ to be faithful. This is a special case of the next result, which indicates how the kernel can be used to characterise faithfulness.

Proposition Characterisation of faithfulness by the (internalised) kernel

Consider a translation $I:L' \to L''$ with (internalised) kernel $Krnl[I] = \langle L', \Lambda[I] \rangle$.

a) For formulae ψ and θ of L', if $I(\psi)=I(\theta)$ then $\Lambda[I] \models \psi \leftrightarrow \theta$.

b) Given a specification $P'=\langle L', G' \rangle$, interpretation $I:P' \to I(P')$ is faithful iff the identifying diagram $\Lambda[I]$ of translation $I:L' \to L''$ consists of consequences of P': $Krnl[I] \subseteq P'$.

Proof.

a) By induction on the structure of formulae of L', preserved by I.

b) Notice that for every $(a_1 \leftrightarrow a_2) \in \Lambda[I]$, $I(a_1)=I(a_2)$; so $\models I(a_1 \leftrightarrow a_2)$.

($\Leftarrow$) Clear: for $\lambda \in \Lambda[I]$, $\varnothing \models I(\lambda)$, so $\lambda \in Cn[P']$.

($\Rightarrow$) Given $I:P' \to I(P')$, we have a surjective $I^*:P' \to I[P']$ with $Krnl[I^*]=Krnl[I]$ and I faithful iff I^* faithful. Thus, it suffices to show that a surjetive $I:P' \to I[P']$ with $Krnl[I] \subseteq P'$ is faithful.

Let A':=Alph(L'). Since mapping I is onto I(A'), we have an injective map $I^\#:I(A') \to A'$ such that the composite $I \circ I^\#$ is the identity on I(A'). This gives a translation $I^\#:I(L') \to L''$ such that $I \circ I^\#(\psi)=\psi$, for $\psi \in Frml(I(L'))$.

Thus, for $\varphi \in Frml(L')$, $I(\varphi)=I \circ I^\# \circ I(\varphi)$, so part (a) yields $\Lambda[I] \models [I^\# \circ I(\varphi) \leftrightarrow \varphi]$.

We claim that $I^\#$ interprets $I[P']=\langle I(L'), I(G') \rangle$ into $\langle L', G' \cup \Lambda[I] \rangle$.

Then, considering a theorem $I(\tau)$ of I[P'], the claim yields $G' \cup \Lambda[I] p I^\#[I(\tau)]$, whence $G' \cup \Lambda[I] \models \tau$ by the above remark.

To see the claim, consider an axiom of I[P']: a sentence $I(\gamma)$ with $\gamma \in G'$. By the remark $\Lambda[I] \models [I^\# \circ I(\gamma) \leftrightarrow \gamma]$, whence $G' \cup \Lambda[I] \models I^\# \circ I(\gamma)$.

{A model-oriented argument can use $I^\#$ to extract from $\mathfrak{K} \in Mod[Krnl[I]]$ a structure $\mathfrak{A}$ for L'' such that $\mathfrak{K} \equiv \mathfrak{A} \downarrow I$.}

QED

4.3.4 Modularisation construction (unsorted)

We shall now examine the Modularisation Construction for (unsorted) interpretations, leaving their extensions to the many-sorted case for later. The Modularisation Construction completes a rectangle of interpretations, and the Modularisation Theorem guarantees that this construction preserves conservativeness, as required for composing implementation steps (see 1.4) and in instantiating parameterised specifications (see 1.5).

The Modularisation Construction deals with the situation where one has

specifications $P=<L_0,G_0>$, $Q=<L_1,G_1>$ and $R=<L_2,G_2>$; as well as an extension $e:P\subseteq Q$, and an interpretation $f:P\rightarrow R$ (see figure 4.4). It completes a rectangle of interpretations by amalgamated sum.

$$\begin{array}{ccc} & Q & \\ e & \cup & \\ P & \xrightarrow[f]{} & R \end{array}$$

Fig. 4.4: Situation for modularisation

The Modularisation Construction is in two stages: one first completes the rectangle of underlying language translations $f:L_1\rightarrow L_2$ and $e:L_0\subseteq L_1$ (see figure 4.5), and then constructs an appropriate specification.

$$\begin{array}{ccc} L_1 & \xrightarrow{g} & L_3 \\ \cup & & \cup \\ L_0 & \xrightarrow[f]{} & L_2 \end{array}$$

Fig. 4.5: Modularisation rectangle of language translations

The unsorted Modularisation Construction for languages proceeds as follows (see figure 4.6). Without loss of generality, we may assume, by resorting to renaming if necessary, that the underlying languages L_1 and L_2 are disjoint: $L_1\cap L_2=\varnothing$. Let $N:=Alph(L_1)-Alph(L_0)$ consist of the new symbols added to L_0 to form L_1. We first construct a new language L_3: we extend L_2 by adding the extra-logical symbols in N (together with their declarations). For instance, the m-ary predicate symbols of L_3 are those of L_2 together with those in N. We now extend the translation $f:L_0\rightarrow L_2$ to a map g from L_1 to the new language L_3 by using the identity on N. We call[1] language L_3 the *amalgamated sum* of L_1 and L_2 under e and f (over L_0), and translation $g:L_1\rightarrow L_3$ the *canonical extension* $e\subseteq[f]$ of $f:L_0\rightarrow L_2$ over e. We have achieved the situation depicted in figure 4.5.

$$g(a) = \begin{cases} f(a) & \text{if } a\in Alph(L_0) \\ a & \text{if } a\in N \end{cases} \qquad L_1 = \left\{ \begin{array}{ccc} N & \xrightarrow{1} & N \\ \cup & & \cup \\ L_0 & \xrightarrow[f]{} & L_2 \end{array} \right\} = L_3$$

Fig. 4.6: Modularisation construction: language and translation

Proposition Properties of the Language Modularisation Construction
Consider an extension $e:L_0\subseteq L_1$ and a translation $f:L_0\rightarrow L_2$. The Language

---

[1] This construction is a special pushout, or amalgamated sum (Ehrich 1982; Goldblatt 1979).

Modularisation Construction yields a language L_3 and a mapping $g:Alph(L_1)\rightarrow Alph(L_3)$ so that:

a) language L_3 extends L_2;

b) g is a language translation from L_1 to L_3 extending $f:L_0\rightarrow L_2$;

c) the only symbols that g maps into L_2 are those of L_0: $.g^{-1}(L_2)=L_0$.

Now, the Modularisation Construction for specifications uses $g:L_1\rightarrow L_3$ to translate the set G_1 of axioms of Q into $g(G_1)$ in language L_3, and constructs specification $S=<L_3,G_3>$ by taking as its set of axioms the union $G_3:=G_2\cup g(G_1)$. (Notice that it would suffice to translate the new axioms in G_1-$Cn[G_0]$.)

Corollary Properties of the Specification Modularisation Construction

Consider specification extension $e:P\subseteq Q$ and interpretation $f:P\rightarrow R$. The Specification Modularisation Construction yields a specification $S=<L_3,G_2\cup g(G_1)>$, such that $S\supseteq R$, and g interprets Q into S.

4.4 The (unsorted) Modularisation Theorem

The Modularisation Theorem guarantees that the Modularisation Construction preserves conservativeness: if $P\leq Q$ then $R\leq S$. We shall now examine the Modularisation Theorem in the context of unsorted interpretations, even though some considerations extend to the many-sorted case, as will be seen later.

We will consider first the special case of Modularity of Extensions in 4.4.1, and then examine Modularity of Interpretations in 4.4.2.

4.4.1 Modularity of (unsorted) extensions

The Modularisation Construction in the special case of extensions yields the union specification. The corresponding version of the Modularisation Theorem, Extension Modularity, asserts that the union construction preserves conservativeness. We shall now examine modularity of (unsorted) extensions, starting with some simple special cases.

A. *Expansive Modularity*

A special case of modularity concerns expansive extensions. In this case, we can resort to model-oriented reasoning to establish preservation of expansiveness.

Proposition Expansive Modularity

Given an orthogonal family of languages L_i, for $i\in I$, let $L_\cap:=\cap_{i\in I}L_i$ and $L^\cup:=\cup_{i\in I}L_i$. Consider a specification $P=<L_\cap,G>$ and a family $P_i=<L_i,G_i>$ of extensions of P, for $i\in I$. Assume that, for some $j\in I$, we have an expansive

extension $P_j \preceq P_k$, for each $k \neq j$ in I. Then, the union specification $P^{\cup}=\langle L^{\cup}, \cup_{i\in I} G_i\rangle$ is an expansive extension of P_j: $P_j \preceq P^{\cup}$.

Proof

Let $K:=I-\{j\}$ and $P^K:=\cup_{k\in K} P_k=\langle \cup_{k\in K} L_k, \cup_{k\in K} G_k\rangle$; and notice that $P^{\cup}=\cup_{i\in I} P_i=P_j \cup P^K$. By the proposition on orthogonal expansiveness in 4.3.1.B, we have an expansive extension $P \preceq P^K$.

Given a model $\mathfrak{A}\in Mod[P_j]$, we will expand it to a structure $\mathfrak{C}\in Mod[P^{\cup}]$. Since $P\subseteq P_j$, the reduct $\mathfrak{A}\downarrow L_{\cap}$ is a model of P; so we have an expansion $\mathfrak{B}\in Mod[P^K]$ of $\mathfrak{A}\downarrow L_{\cap}$. Thus, we have a family $\{\mathfrak{A},\mathfrak{B}\}$ of structures for the orthogonal languages L_j and $L^K:=\cup_{k\in K} L_k$, with common reduct $\mathfrak{A}\downarrow L_{\cap}$ to $L_{\cap}=L_j \cap L^K$. By the lemma on joint expandabilitiy for orthogonal family (see 4.3.1.B), the family $\{\mathfrak{A},\mathfrak{B}\}$ has a common expansion $\mathfrak{C}$.

Since $\mathfrak{C}$ expands both $\mathfrak{A}\in Mod[P_j]$ and $\mathfrak{B}\in Mod[P^K]$, $\mathfrak{C}\in Mod[P_j \cup P^K]=Mod[P^{\cup}]$.

QED

Expansiveness is a sufficient condition for conservativeness (see 3.4); but, unfortunately, it is not necessary. So, we cannot immediately conclude Extension Modularity from Expansive Modularity.

B. *A first proof attempt (of extension modularity) and its difficulties*

Let us now sketch what might be a first approach towards a proof of the Modularisation Theorem for extensions and some of the difficulties encountered.[1]

We have that Q is a conservative extension of P and we wish to prove that $R\leq S$. For this purpose, we must take an arbitrary sentence τ of L_2, such that $S\models \tau$ and establish $R\models \tau$. The point is that we have to rely on the assumed conservativeness $P\leq Q$, which is the idea behind this proof attempt. We might proceed as follows.

0. $S\models \sigma$	with $\sigma\in Sent(L_0)$	σ in $L_0\subseteq L_2$ {special case!}
1. $Q\cup R\models \sigma$		since $S=Q\cup R$
2. $Q\models \sigma$	with $\sigma\in Sent(L_0)$	by ignoring R {why?}
3. $P\models \sigma$		because $P\leq Q$
4. $R\models \sigma$		because $P\subseteq R$

Now, there are two objections to this proof attempt:

1. it deals only with the special case of sentences in $L_0\subseteq L_2$;
2. step 2 is 'justifed' by "ignoring R" (and how is this justified?).

---

[1] The reader not interested in this motivation may proceed directly to C. Interpolation properties.

The point is that we must rely on the assumed conservativeness $P \leq Q$, which is the idea behind the above proof attempt. We will overcome these difficulties by reducing Extension Modularity to a composition of two special cases: Language Modularity followed by Axiom Modularity. The former will reduce $\tau \in Sent(L_2)$ to be $\sigma \in Sent(L_0)$, thus overcoming the special assumption (in step 0), and the latter will justify "ignoring R" (in step 2).

C. *Interpolation properties*

We shall now examine some interpolation properties of first-order logic which will be used to prove some versions of Modularity of Extensions.

An important property of first-order logic is the so called Craig Interpolation Property, which appears in a few versions in the literature (Shoenfield 1967; Chang and Keisler 1973). Such interpolation properties enable the decomposition of derivations involving formulae in distinct languages by interpolating formulae with the common extra-logical symbols. A simple version of the *Craig Interpolation Lemma* is as follows.

Proposition Simple Craig Interpolation (Chang and Keisler 1973, p. 84)

Given sentences σ of L^* and τ of $L^\#$, if $\tau \in Cn(\sigma)$, then there exists an *interpolant sentence* ρ of $L^* \cap L^\#$, such that $\sigma \models \rho$ and $\rho \models \tau$.

A model-theoretic formulation is "whenever $Mod(\sigma) \subseteq Mod(\tau)$ with $\sigma \in Sent(L^*)$ and $\tau \in Sent(L^\#)$, there exists $\rho \in Sent(L^* \cap L^\#)$ such that $Mod(\sigma) \subseteq Mod(\rho) \subseteq Mod(\tau)$". A variant of the Craig Interpolation Lemma is the so called *Split Interpolation* version (Rodenburg and van Glabbeek 1988). This is close to the following usual formulation of the *Craig-Robinson Interpolation Lemma* (Shoenfield 1967, p. 80). Consider specifications $P^* = \langle L^*, G^* \rangle$ and $P^\# = \langle L^\#, G^\# \rangle$ with union $P^* \cup P^\#$. If $(\phi \rightarrow \psi) \in Cn[P^* \cup P^\#]$, for formulae $\phi \in Frml(L^*)$ and $\psi \in Frml(L^\#)$, then there exists an *interpolant formula* $\theta \in Frml(L^* \cap L^\#)$, such that $(\phi \rightarrow \theta) \in Cn[P^*]$ and $(\theta \rightarrow \psi) \in Cn[P^\#]$.

In classical first-order logic, these versions of interpolation are interderivable, because of the Compactness and Deduction Theorems. Any such version can be applied to the above proof attempt to overcome the difficulties encountered. We prefer, however, to follow an alternative route, which will clarify the roles played by these interpolation properties in guaranteeing modularity of extensions.

D. *Language, Axiom and Extension Modularity*

We shall now establish modularity of (unsorted) extensions, starting with some simple special cases. One can view an extension as being constructed by a two step procedure: first add new symbols, then add new axioms (see

3.4). Two simple properties, which turn out to be special cases of Modularity of Extensions, refer to these two steps: modularity with respect to language and axioms.

Language modularity concerns the addition of new symbols to both languages of an extension; it guarantees that such an addition preserves conservativeness (see figure 4.7). This property of Language Modularity should not be confused with the more familiar property of symbol extensions (see 3.4). The latter asserts that the addition of new symbols produces a conservative extension; whereas what this property asserts is that such an addition preserves conservativeness.

$$L \cap L'' \subseteq L' \Rightarrow \begin{bmatrix} < L'',G'' > & \subseteq & < L \cup L'',G'' > \\ V & \Rightarrow & V \\ < L',G' > & \subseteq & < L \cup L',G' > \end{bmatrix}$$

Fig. 4.7: Language Modularity: addition of new symbols

Lemma LM: Language Modularity

Consider a conservative extension $P'=<L',G'>\leq<L'',G''>=P''$. Then, for each language L, compatible with L'', such that $L \cap L'' \subseteq L'$, we have a conservative extension $<L' \cup L,G'>\leq<L'' \cup L,G''>$.

Proof

First, notice that $L'' \cap (L' \cup L)=(L'' \cap L') \cup (L'' \cap L)=L' \cup (L'' \cap L)=L'$.

Now, consider a sentence τ of $L' \cup L$ such that $G'' \models \tau$ (in $L'' \cup L$).

By Compactness[1], there exists a sentence $\sigma \in$ Sent(L'') such that

(i) $G'' \models \sigma$ and (ii) $\sigma \models \tau$.

The Craig Interpolation Lemma applied to (ii) yields an interpolant sentence ρ of $L'' \cap (L' \cup L)=L'$, such that

(iii) $\sigma \models \rho$ and (iv) $\rho \models \tau$.

From (i) and (iii) we have $G'' \models \rho$ with $\rho \in$ Sent(L). Thus, $<L',G'>\leq<L'',G''>$ yields $G' \models \rho$. Hence, from (iv), we have $G' \models \tau$.

QED

In fact, Language Modularity turns out to be equivalent to Simple Craig Interpolation, a remark that provides a reformulation of the latter as preservation of conservativeness under addition of new symbols. Also, it is not difficult to see that Language Modularity is a special case of Modularity of Extensions. Indeed, given $<L',G'>\leq<L'',G''>$ with $L \cap L'' \subseteq L'$, we

[1] Notice that this formulation of (simple) compactness can be viewed as an interpolation property: σ is a finite conjunction of sentences of G''.

also have $\langle L',G'\rangle \subseteq \langle L'\cup L,G'\rangle$ with $L''\cap(L'\cup L)=L'$; hence Extension Modularity yields $\langle L'\cup L,G'\rangle \leq \langle L''\cup L'\cup L,G'\cup G''\rangle \cong \langle L''\cup L,G''\rangle$, as required by LM.

The second property, axiom modularity, concerns the addition to both specifications of a set of new sentences of the smaller language. It asserts that such addition preserves conservativeness (see figure 4.8).

$$H \subseteq Sent(L') \Rightarrow \begin{bmatrix} \langle L'',G''\rangle & \subseteq & \langle L'',H\cup G''\rangle \\ V & \Rightarrow & V \\ \langle L',G'\rangle & \subseteq & \langle L',H\cup G'\rangle \end{bmatrix}$$

Fig. 4.8: Axiom Modularity: addition of new sentences

Lemma AM: Axiom Modularity

Consider a conservative extension $P'=\langle L',G'\rangle \leq \langle L'',G''\rangle = P''$. Then, for every set H of sentences of language L', we have a conservative extension $\langle L',G'\cup H\rangle \leq \langle L'',G''\cup H\rangle$.

Proof

{This is a special case of the corollary on characterisation of faithfulness (for unsorted specifications) in 4.2.3. We present a direct proof to note some connections with interpolation.}

Consider a sentence σ of L' such that $G''\cup H\models\sigma$.

By Compactness[1], there exists a sentence $\theta\in Sent(L')$, such that

(i) $H\models\theta$ and (ii) $G''\cup\{\theta\}\models\sigma$.

From (ii) we have a sentence[2] χ of L', such that

(iii) $G''\models\chi$ and (iv) $\{\chi\}\cup\{\theta\}\models\sigma$.

From (iii), since $\langle L',G'\rangle \leq \langle L'',G''\rangle$ and $\chi\in Sent(L')$, we have $G'\models\chi$. Thus, by the Cut Rule, (iv) yields $G'\cup\{\theta\}\models\sigma$, whence $G'\cup H\models\sigma$, in view of (i).

QED

It is easy to see that Axiom Modularity is a special case of Modularity of Extensions. Indeed, given $\langle L',G'\rangle \leq \langle L'',G''\rangle$ and a set $H\subseteq Sent(L')$, we also have $\langle L',G'\rangle \subseteq \langle L',G'\cup H\rangle$; hence Modularity of Extensions yields, as required by AM: $\langle L',G'\cup H\rangle \leq \langle L''\cup L',G''\cup G'\cup H\rangle \cong \langle L'',G''\cup H\rangle$.

The preceding remarks indicate that both Language and Axiom Modularity follow from Extension Modularity. We now derive the latter from them, thereby establishing Modularity of Extensions.

---

[1] Notice that this formulation of (distributed) compactness can be viewed as an interpolation property: θ is a finite conjunction of sentences of H.

[2] Notice that this is connective-free formulation of the Deduction Theorem: χ can be taken as $(\theta\rightarrow\sigma)$.

Proposition EM: Modularity of (unsorted) Extensions

Given sub-languages L_1 and L_2 of L, let $L_0=L_1\cap L_2$. Consider a specification $P=<L_0,G_0>$, with extensions $Q=<L_1,G_1>$ and $R=<L_2,G_2>$. If Q is a conservative extension of P ($P\leq Q$), then the union specification $S:=<L_1\cup L_2,G_1\cup G_2>$ is a conservative extension of R: $R\leq S$.

Proof (see figure 4.9)

By Language Modularity, we have $<L_2,G_0>=<L_0\cup L_2,G_0>\leq<L_1\cup L_2,G_1>$. Thus, Axiom Modularity yields $<L_2,G_2>\equiv<L_2,G_0\cup G_2>\leq<L_1\cup L_2,G_1\cup G_2>$, since $G_2\subseteq Sent(L_2)$.

QED

$$\begin{array}{ccccc} \overbrace{<L_1,G_1>}^{Q} & \subseteq & <L_1\cup L_2,G_1> & \subseteq & \overbrace{<L_1\cup L_2,G_1\cup G_2>}^{S=Q\cup R} \\ \vee & \overset{LM}{\Rightarrow} & \vee & \overset{AM}{\Rightarrow} & \vee \\ \underbrace{<L_0,G_0>}_{P} & \subseteq & <\underbrace{L_0\cup L_2}_{L_2},G_0> & \subseteq & <L_2,G_0\cup G_2>\equiv R \end{array}$$

Fig. 4.9: Proof of Modularity of Extensions: LM&AM⇒EM

E. *General Modularity of Orthogonal Extensions*

We can now generalise modularity of extensions to specifications over orthogonal families of languages in the spirit of Expansive Modularity.

We first establish that union of an orthogonal family preserves conservativeness, a result akin to orthogonal expansiveness in 4.3.1.B.

Lemma Orthogonal conservativeness

Given an orthogonal family of languages L_i, for $i\in I$, let $L_\cap:=\cap_{i\in I}L_i$.

Consider a specification $P=<L_\cap,G>$ and a family $P_i=<L_i,G_i>$ of extensions of P, for $i\in I$. Assume that, for every $i\in I$, we have a conservative extension $P\leq P_i$. Then, the union specification $P^\cup=\cup_{i\in I}P_i$ is a conservative extension of P: $P\leq P^\cup$.

Proof

Consider $\sigma\in Sent(L_k)$ such that $\sigma\in Cn[P^\cup]$.

Then, by compactness, there exists a finite $F\subseteq I$ such that $\sigma\in Cn[\cup_{i\in F}P_i]$.

Notice that we may assume $F\neq\emptyset$. We claim that $P\leq\cup_{i\in F}P_i$. Then $\sigma\in Cn[P]$.

The claim follows from Extension Modularity by induction.

The basis, with $|F|=1$, being trivial, consider the inductive step.

Given F with $|F|>1$, choose $f\in F$ and set $F':=F-\{f\}\subset F$. The inductive hypothesis gives $P\leq\cup_{i\in F'}P_i$. Since $L_f\cap(\cup_{i\in F'}L_i)=\cup_{i\in F'}(L_f\cap L_i)\subseteq L_\cap$, Extension Modularity yields $P_f\leq P_f\cup(\cup_{i\in F'}P_i)=\cup_{i\in F}P_i$. Hence $P\leq P_f\leq\cup_{i\in F}P_i$.

Therefore $P\leq P^\cup$.

QED

We can now establish the generalisation of Extension Modularity to specifications over an orthogonal family guaranteeing preservation of conservativeness under such union.

Theorem Orthogonal Extension Modularity

Given an orthogonal family of languages L_i, for $i\in I$, let $L_\cap:=\cap_{i\in I}L_i$.
Consider a specification $P=<L_\cap,G>$ and a family $P_i=<L_i,G_i>$ of extensions of P, for $i\in I$. Assume that, for some $j\in I$, we have a conservative extension $P_j\leq P_k$, for each $k\neq j$ in I. Then, the union specification $P^\cup=\cup_{i\in I}P_i$ is a conservative extension of P_j: $P_j\leq P^\cup$.

Proof

Let $K:=I-\{j\}$. By the preceding lemma, we have a conservative extension $P\leq P^K$, where $P^K:=\cup_{k\in K}P_k=<\cup_{k\in K}L_k,\cup_{k\in K}G_k>$.
Notice that $P^\cup=\cup_{i\in I}P_i=P_j\cup P^K$ and $L_j\cap(\cup_{k\in K}L_k)=\cup_{k\in K}(L_j\cap L_k)\subseteq L_\cap$.
Thus, Extension Modularity yields $P_j\leq P_j\cup P^K=P^\cup$.

QED

4.4.2 Modularity of (unsorted) interpretations

We shall now consider the case of Modularity of Interpretations, rather than extensions, aiming at establishing the Modularisation Theorem for (unsorted) specifications.

We will examine some cases where the extension $P\leq Q$ has some special form and then consider a first - unsuccessful - attempt at reducing modularity of interpretations to that of extensions, which will point out the difficulty to be overcome by the successful reduction.

A. *Modularity with special syntactical forms*

We shall first examine some special cases, where the given extension $P\leq Q$ has some known syntactical form. The proof of the Modularisation Theorem for such cases is quite straightforward, as we shall see.

Consider first the case where the extension from P to Q is by definitions: of predicates (see 3.5.1) and operations (see 3.6.1). Let us re-examine again the Modularisation Construction given in 4.3.4. If the extension $P\subseteq Q$ happens to be by definitions, then G_1 consists of G_0 with some new axioms defining the symbols in N in terms of those of L_0. It is easy to see that the

translation via g of each such axiom is still a defining axiom: of a new symbol of L_3-L_2 in terms of those of L_2 Hence, the Modularisation Construction yields S as an extension by definitions of R.

Since an extension by definitions is known to be a special case of conservative extension; this is indeed a special version of the Modularisation Theorem: with a stronger hypothesis, but also yielding a more informative conclusion.

Notice that the argument outlined above relies heavily on the special syntactical form of the axioms added in extending P to Q. It carries over to other kinds of conservative extensions with such special syntactical form: addition of predicates via constraints or of inductive predicates (see 3.5.2 and 3.5.3) as well as addition of operations via constraints (see 3.6.2). It does not, however, appear to extend to the general case, when the new axioms are not guaranteed to have any such special form.

B. *A first reduction attempt and its difficulties*

We now sketch what might be a first attempt towards a proof of (the general case of) the Modularisation Theorem and some of the difficulties encountered in trying to derive it from modularity of extensions.

Let us outline an approach for reducing Modularity of Interpretations to that of Extensions. We can start by constructing the language postimages: $f[P]=\langle f(L_0),f(G_0)\rangle$ and $g[Q]=\langle g(L_1),g(G_1)\rangle$. We then have the situation in figure 4.10. By properties of the Language Modularisation Construction, we have $g^{-1}(L_2)=L_0$, so $L_2 \cap g(L_1)=f(L_0)$. Thus, if we had $f[P] \leq g[Q]$, we could apply Extension Modularity to conclude $R \equiv f[P] \cup R \leq g[Q] \cup R \equiv S$. So, this reduces the problem to $f[P] \leq g[Q]$.

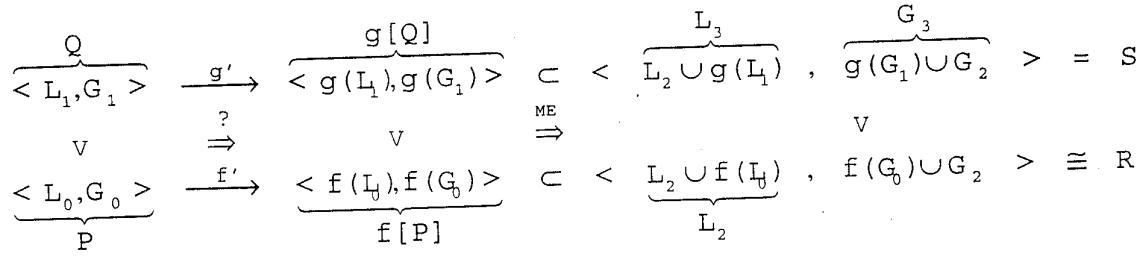

Fig. 4.10: Reducing modularity of interpretations to extensions

We now wish to establish $f[P] \leq g[Q]$: given $\tau \in g(L_1)$ such that $g[Q] \models \tau$, we must show that $f[P] \models \tau$. For this, we must rely on the assumed conservativeness $P \leq Q$. This is the idea of this reduction attempt. We might

proceed as follows[1].

0. $g[Q] \models f(\sigma)$	with $\sigma \in Sent(L_0)$	assumption
1. $g[Q] \models g(\sigma)$		because f and g agree on L_0
2. $Q \models \sigma$	with $\sigma \in Sent(L_0)$	by cancelling g {wow!}
3. $P \models \sigma$		because $P \leq Q$
4. $f[P] \models f(\sigma)$		because $f: P \rightarrow f[P]$

There is an objectionable step in this proof attempt:

step 1 is 'justifed' by "cancelling g".

Now, cancelling g would be acceptable if g happened to be faithful. Note that, in view of the Specification Modularisation Construction, g would be faithful if f happened to be so. But this is an unwarranted assumption, which would mar the intended applications of the result to composing implementations and to parameter instantiation, as indicated in 1.4 and 1.5.

The point is that we must rely on the assumed conservativeness $P \leq Q$, which is the idea behind the above reduction attempt. It would work if f (and g) happened to be faithful; otherwise we have an unjustified step. This can be overcome by resorting to the internalisation techniques in 4.3.3: we replace the given specifications P and Q by stronger ones, without affecting conservativeness or interpretability, so that both f and g become faithful, thereby justifying the above (previously) objectionable step. We will still start from a conservative extension, but now have faithful interpretations, so we will be able to establish $f[P] \leq g[Q]$.

C. *Reduction of Modularity: Interpretations to Extensions*

We will now establish (the general case of) the Modularisation Theorem by reducing it to Modularity of Extensions via the internalised kernel.

We must rely on the assumed conservativeness $P \leq Q$. The key idea is to replace the given given P and Q in figure 4.10 by their kernel extensions $P \cup Krnl[f]$ and $Q \cup Krnl[f]$. This will not affect conservativeness - by Axiom Modularity - or interpretability. Also, in view of the Modularisation Construction, g will become faithful[2].

Theorem Modularity of (unsorted) Interpretations

Consider specifications $P=\langle L_0, G_0 \rangle$, $Q=\langle L_1, G_1 \rangle$ and $R=\langle L_2, G_2 \rangle$; as well as an extension $e: P \subseteq Q$, and an interpretation $f: P \rightarrow R$. Let $S=\langle L_3, G_2 \cup g(G_1) \rangle$ be the

---

[1] The reader not interested in this motivation may proceed directly to C. Reduction of Modularity: Interpretations to Extensions.

[2] This will overcome the difficulty with "cancelling g" in step 1 in the reduction attempt in 4.4.1.B

specification yielded by the Specification Modularisation Construction. If Q is a conservative extension of P ($P \leq Q$), then S is a conservative extension of R: $R \leq S$.

Proof (see figure 4.10)

We consider the language postimages $f[P]=\langle f(L_0), f(G_0)\rangle$ and $g[Q]=\langle g(L_1), g(G_1)\rangle$, and claim that $f[P] \leq g[Q]$. We will then have the situation in figure 4.10. By the proposition on properties of the Language Modularisation Construction (in 4.3.4), $g^{-1}(L_2)=L_0$, so $L_2 \cap g(L_1)=f(L_0)$. So, we can apply Extension Modularity (see 4.4.2.B) to conclude $R \cong f[P] \cup R \leq g[Q] \cup R \cong S$.

To establish the claim that $f[P] \leq g[Q]$, we extend $P=\langle L_0, G_0\rangle$ and $Q=\langle L_1, G_1\rangle$ by the (internalised) kernel $Krnl[f]=\langle L_0, \Lambda[f]\rangle$ obtaining $P \cup Krnl[f] := \langle L_0, G_0 \cup \Lambda[f]\rangle$ and $Q \cup Krnl[f] := \langle L_1, G_1 \cup \Lambda[f]\rangle$ and show that $P \cup Krnl[f] \leq Q \cup Krnl[f]$ and that g interprets $Q \cup Krnl[f]$ faithfully into g[Q].

Since $\Lambda[f] \subseteq Sent(L_0)$ and $P \leq Q$, Axiom Modularity yields $P \cup Krnl[f] \leq Q \cup Krnl[f]$. Also, by the Language Modularisation Construction in 4.3.4, g extends f by the identity. Thus, we have $\Lambda[g] \subseteq Cn[Krnl[f]]$, whence $Q \cup Krnl[f] \cong Q \cup Krnl[g]$. Thus, by the proposition on characterisation of faithfulness by (internalised) kernel in 4.3.3, we have the faithfulness of $g: Q \cup Krnl[f] \rightarrow Q$.

Now, given $\sigma \in Sent(L_0)$, such that $g[Q] \models f(\sigma)$, we have successively:

$g[Q] \models g(\sigma)$, $Q \cup Krnl[f] \models \sigma$, $Q \models \sigma$, $P \models \sigma$, $f[P] \models f(\sigma)$.

QED

4.5 Variations of (unsorted) interpretations

We have been concentrating on symbol translations, which map symbols to symbols. Other possibilities also occur in the literature: mapping symbols to formulae, relativisation of quantifiers and translating equality (Enderton 1972; Shoenfield 1967; Turski and Maibaum 1987). We shall now examine them, as well as translations of variables.

These relaxations of translation affect only the translation of terms and formulae; the concept of interpretation still requires preservation of theorems. Thanks to the decomposition in 4.3.2.C, we can isolate their effects on each symbol, without loss of generality.

4.5.1 Translation of predicates

A version of translation often encountered in the literature assigns to a predicate symbol a formula, rather than a predicate symbol (Enderton 1972, p. 158). As might be expected, such a predicate translation can be replaced by a symbol interpretation into an extension by definition.

Let us concentrate on examining the case of a predicate symbol being

mapped to a formula. Consider (unsorted) languages L' and L"; a *formula translation* $I:L' \rightarrow L''$ *for* m-ary *predicate* symbol r of L' assigns to r a formula ρ of L" with free variables $v_1,\dots,v_m$. The idea is that atomic formula $r(t_1,\dots,t_m)$ of L' is to be translated to $\rho(t_1,\dots,t_m)$ (see, e. g. Enderton 1972, p. 160). This is how it affects the translation of formulae.

For a simple example, consider the introduction of a new predicate symbol r into specification $P=<L,G>$ with defining axiom $\delta(r\backslash\rho)$ of the form $\forall v_1 \dots \forall v_m[r(v_1,\dots,v_m)\leftrightarrow\rho]$ with ρ a formula of L, as in 3.5.1. We then have an expansive extension $P^e=<L\cup\{r\},G\cup\{\delta(r\backslash\rho)\}>$, with eliminability. The latter means that we have a function e mapping formulae of $L\cup\{r\}$ back into L, so that $\delta(r\backslash\rho)\models[\psi\leftrightarrow e(\psi)]$. We thus have a a formula translation $e:L\cup\{r\}\rightarrow L$ for predicate symbol r, which is a faithful interpretation of P^e into P. Notice that e acts identically on the symbols of sub-language L.

This example suggests how we can reduce a formula interpretation to a symbol interpretation in the case of a predicate: the sentence $\forall v_1 \dots \forall v_m[r(v_1,\dots,v_m)\leftrightarrow\rho]$ of L" has the form of a definition of predicate symbol $r\in$ Prd(L') in terms of $\rho\in$Frml(L") (see 3.5.1). Thus, the next result - illustrated in figure 4.11 - is not surprising.

Proposition Formula vs. symbol interpretation (for a predicate)

Given a formula translation $I:L' \rightarrow L''$ assigning to predicate symbol r of L' formula ρ of L", let $\delta(r\backslash\rho)$ be the sentence $\forall v_1 \dots \forall v_m[r(v_1,\dots,v_m)\leftrightarrow\rho]$.

a) Specification $<L''\cup\{r\},\{\delta(r\backslash\rho)\}>$ is an extension by definition of $Trv(L'')=<L'',\varnothing>$ where I interprets $Trv(L')=<L',\varnothing>$.

b) There exists a symbol interpretation $I^*:<L',\varnothing>\rightarrow<L''\cup\{r\},\{\delta(r\backslash\rho)\}>$ equivalent to the composite $I':Trv(L')\rightarrow Trv(L'')\subseteq<L''\cup\{r\},\Delta[I]>$ in the following sense.

(i) For every $\varphi\in$Frml(L'): $\delta(r\backslash\rho)\models I(\varphi)\leftrightarrow I^*(\varphi)$.

(ii) Given specifications $P'=<L',G'>$ and $P''=<L'',G''>$, I interprets P' into P" iff I^* interprets P' into the extension $P^*=<L''\cup\{r\},G''\cup\{\delta(r\backslash\rho)\}> \triangleright P''$.

$$\begin{array}{ccc} <L',G'> & \xrightarrow{\ I\ } & <L'',G''> \\ \| & & \Delta \\ <L',G'> & \xrightarrow{\ I^*\ } & <L''\cup\{r\}\{\delta(r\backslash r)\}\cup G''> \end{array}$$

Fig. 4.11: Reduction of formula to symbol translation for predicate

The content of this result is that we can always reduce a formula interpretation for a predicate to a symbol interpretation with the same effect. For instance, if we have another interpretation $J:P''\rightarrow P$, we can

replace the composite $I;J:P' \to P$ by $I^*;J^*:P' \to P$, where J^* is the composition of the given $J:P'' \to P$ with the faithful $e:P^* \to P''$ which eliminates r back to ρ. This is illustrated in figure 4.12.

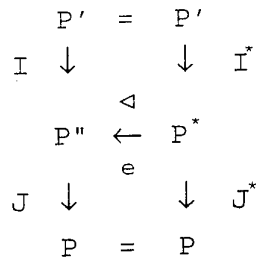

Fig. 4.12: Reduction of formula to symbol interpretations

4.5.2 Translation of operations

Another version of translation often encountered in the literature assigns to an operation symbol a formula, rather than an operation symbol (Enderton 1972, p. 158). As might be expected, such an assignment involves some restrictions - like those involved in definitions of operations - in order to exhibit the desired behaviour.

Formula interpretation for operation can be illustrated by considering the specifications for Booleans in 3.10 (see Spec 3.1 and 3.2). One can assign to binary predicate symbol less? of the language of BOOL_EXT_NEG&LESS the formula $x \approx fl \wedge y \approx tr$ of the language of BOOL.

Let us concentrate on the case of an operation symbol being mapped to a formula. Consider (unsorted) languages L' and L"; a *formula translation* $I:L' \to L''$ *for* n-ary *operation* symbol f of L' assigns to f a formula θ of L" with free variables $x_1,\ldots,x_n,y$. The idea is that atomic formula $f(t_1,\ldots,t_n) \approx t$ of L' is to be translated to $\theta(t_1,\ldots,t_n,t)$ (see, e. g. Enderton 1972, p. 160). This takes care of the translation of formulae, even if not of terms.

If we adapt the idea of the previous reduction, we obtain the sentence $\forall x_1 \ldots \forall x_n \forall y[f(x_1,\ldots,x_n) \approx y \leftrightarrow \theta]$, which is the form of a definition of operation symbol $f \in Opr(L')$ in terms of $\theta \in Frml(L'')$ (see 3.6.1). This indicates that some precautions are necessary. The requirements to be imposed on such a formula translation for an operation have two, related, goals.

We wish to preserve induction of structures; for this the realisation of formula θ in a structure for L" must be the graph of a function.

We wish such translation to preserve logical validities; and the translation of the valid sentence $\forall x_1 \ldots \forall x_n \exists! y f(x_1,\ldots,x_n) \approx y$ is - equivalent to - the conjunction of the existence and uniqueness conditions for θ (see 3.6.1).

We are thus led to restricting such formula translations for operations -

in the spirit of the interpretation of a language into a theory (Shoenfield 1967, p. 61) - as follows. Given a specification $P''=<L'',G''>$, we say that formula translation $I:L'\rightarrow L''$, assigning to operation symbol f of L' formula θ of L", is a *formula interpretation* $I:<L',\varnothing>\rightarrow<L'',G''>$ *for operation* symbol f of L' iff the existence $\exists(\theta)$ and uniqueness $!(\theta)$ conditions for θ are in Cn[P"]. Recall, from 3.6.1, that these conditions are expressed by $\forall x_1 \ldots \forall x_n \exists y \theta$ and $\forall x_1 \ldots \forall x_n \forall y' \forall y''\{[\theta(x_1,\ldots,x_n,y') \wedge \theta(x_1,\ldots,x_n,y'')] \rightarrow y' \approx y''\}$, respectively.

With these requirements we achieve the desired goals. But, as in the case of predicates, such formula interpretations can be reduced to symbol interpretations without any loss.

Proposition Formula vs. symbol interpretation (for an operation)

Given a formula translation $I:L'\rightarrow L''$ assigning to operation symbol f of L' formula θ of L", let $\delta(f\backslash\theta)$ be $\forall x_1 \ldots \forall x_n \forall y[f(x_1,\ldots,x_n) \approx y \leftrightarrow \theta]$. Given specification $P''=<L'',G''>$, consider its extension $P^*=<L'' \cup \{f\}, G'' \cup \{\delta(f\backslash\theta)\}>$.

a) The following are equivalent for a specification $P'=<L',G'>$.

(i) I is a formula interpretation $I:P'\rightarrow P''$ for operation symbol f.

(ii) P^* is a conservative extension of P".

(iii) P^* is an extension by definition of P".

b) If $P'' \models \exists(\theta) \wedge !(\theta)$, then there exists a symbol interpretation $I^*:Trv(L')\rightarrow P^*$ equivalent to the composite $I':Trv(L')\rightarrow P'' \subseteq P^*$ in the following sense.

(i) For every $\varphi \in Frml(L')$: $\delta(f\backslash\theta) \models I(\varphi) \leftrightarrow I^*(\varphi)$.

(ii) Given a specification $P'=<L',G'>$, I interprets P' into P" iff I^* interprets P' into P^*.

4.5.3 Translation of variables

As mentioned in 4.2.1, in the unsorted case we do not have to translate the variables, but we may. Since translation of variables appears naturally in the many-sorted case, we shall now examine what is involved in translating variables. By an argument similar to that of decomposition of translations in 4.3.2.C, we can treat this case in isolation.

Consider (unsorted) languages L' and L" with sets of variables $V'=Var(L')$ and $V''=Var(L'')$. A *translation of variables* from L' to L" would be a function $I:V'\rightarrow V''$ mapping each variable v' of L' to a corresponding variable v" of L". The idea is that a term $t \in Trm(L')$ - respectively, formula $\varphi \in Frml(L')$ - is translated by simply replacing all occurrences of variables v' in V' by $I(v') \in V''$. Thus, translation of terms and formulae is straightforward.

We are concentrating on translation of variables, leaving the other symbols untouched, so Alph(L')=Alph(L"). Thus, each structure $\mathfrak{M}$ for source language L' is already a structure for L". Now, given an assignment a:V"$\rightarrow$M of values in the universe of structure $\mathfrak{M}$ for the variables of L', we can compose it with the translation I:V'$\rightarrow$V", to obtain a composite assignment I;a:V'$\rightarrow$M for the variables of L", as illustrated in figure 4.13.

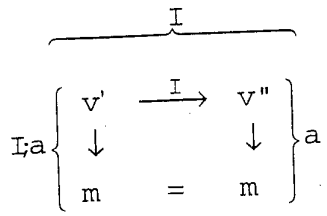

Fig. 4.13: Assignment induced by translation of variables

Variables play a somewhat subsidiary role in safisfaction of sentences in structures; but, for formulae, we would wish that a formula and its translation "express the same thing". For this, some precautions concerning identification of variables are necessary. The requirements to be imposed on such translations of variables have two - related - goals, namely

we wish to preserve the Translation Connection, in the now natural form: $\mathfrak{M}\models$ I(φ) [a"] iff $\mathfrak{M}\models\varphi$ [I;a"]; and

we wish such a translation to preserve logical consistency: a non-contradictory formula φ should not be translated to a contradiction.

Now, assume that two distinct variables u' and w' of L' get translated to the same variable v" of L". Then, formula $\exists$u'($\neg$u'$\approx$w') is translated to $\exists$v"($\neg$v"$\approx$v"), undermining both goals.

It is thus in general wise to restrict our attention to one-to-one translations of variables. A *faithful translation of variables* from L' to L" is an injective function I:V'$\rightarrow$V", mapping distinct variables of L' to distinct variables of L".

The reason for the name 'faithful' translation of variables is that a translation of variables I:V'$\rightarrow$V" interprets faithfully Trv(L')=<L',$\varnothing$> into Trv(L")=<L",$\varnothing$> iff it is injective. For such injective translations of variables, we have the desired goals. But, as in the case of predicates, they can be reduced to symbol translations without any loss, by suitable renamings.

4.5.4 Other cases: relativisation and equality

We shall now briefly examine two other versions of translations that are important for implementations. One of them deals with relativising the range of variables and the other one with translation of equality. As might be expected, these versions involve some restrictions. The effects of these variants are less local than the preceding ones. They can be reduced to

introduction of sorts, subsort and quotient sort (see 3.8), as will be seen later on when we consider the many-sorted case (in 4.9.1 and 4.9.2).

Our example of implementing sets of naturals by sequences of naturals in 4.1 indicates why they are useful. The relativisation predicate will restrict quantification to the sequences that represent sets - those without repetitions - and equality of sets will be translated to sequences having the same elements, regardless of their place of occurrence.

A. *Translation with relativisation*

A version of translation/interpretation often encountered involves relativisation predicates (Enderton 1972, p. 157-161; Shoenfield 1967, p. 61; Turski and Maibaum 1987, p. 82).

Such a translation I:L'$\rightarrow$L" with a relativisation predicate provides a unary predicate symbol r of L". The idea is that r represents within L" the universe of L', over which quantifiers of L' range. The effect of this in translating is felt only in the quantified formulae of L': the quantifiers are relativised by replacing each part $\forall v\ldots$ by $\forall v[r(v)\rightarrow\ldots]$ and $\exists v\ldots$ by $\exists v[r(v)\wedge\ldots]$ (Enderton 1972, p. 157-161). (One sometimes also prefixes the result by $r(v_1)\wedge\ldots\wedge r(v_m)$, where $v_1,\ldots,v_m$ are its free variables (see, e. g. Shoenfield 1967, p. 62; Turski and Maibaum 1987, p. 83).)

As in the case of a formula translation of operations, some care is necessary. The requirements to be imposed on such a translation with a relativisation predicate have two - related - aims.

From a model-oriented standpoint, we wish to induce structures taking the extension of r as the universe of the structure for L'; for this the realisation of the relativisation predicate r in a structure for L" must be nonempty. It must also be closed under the realisation of each operation symbol.

From a property-oriented viewpoint, we wish such a translation to preserve logical validities; and the relativisation of the valid sentence $\exists v v\approx v$ is equivalent to $\exists v r(v)$. Also, the relativisation of the valid sentence $\forall x_1\ldots\forall x_n\exists y f'(x_1,\ldots,x_n)\approx y$ expresses the above closure requirement for operation symbol I(f').

We are thus led to restricting such translations with relativisation predicates - once again as interpretations of languages into theories (Shoenfield 1967, p. 61) - as follows. We consider an interpretation I:Trv(L')$\rightarrow$P" with relativisation predicate r just in case the sentences $\exists v r(v)$ and $\forall x_1\ldots\forall x_n\exists y f'(x_1,\ldots,x_n)\approx y$, for each source operation symbol f', are consequences of P". With these requirements we achieve the desired goals. Such interpretations with relativisation predicates can be reduced to symbol interpretations into subsorts (see 3.8.2). We shall come back to this point when examining many-sorted interpretations in 4.9.1.

B. *Translation of equality*

A variant of translation/interpretation sometimes encountered involves translation of equality (Turski and Maibaum 1987, p. 163). This is motivated by the fact that, as in our example in 4.1, several target objects may represent the same source object.

Such a translation $I:L' \rightarrow L''$ of equality provides a binary predicate symbol q of L". The idea is that q represents within L" the identity on the universe of L'. The effect of this in translating is felt only in the equality formulae: an atomic formula $t \approx t'$ of L' is translated to $q(t,t')$ (see, e. g. Turski and Maibaum 1987, p. 163).

As in the case of translation with a relativisation predicate, some precautions are needed for proper behaviour.

From a model-oriented standpoint, to maintain the idea that q represents within L" the identity on the universe of L', we wish to take $\mathfrak{M}' = \mathfrak{M}''/\mathfrak{M}''[q]$; for this the realisation of q in a structure $\mathfrak{M}''$ for L" must be an equivalence relation. It must also be a congruence if we wish the natural projection onto the quotient to be a homomorphism.

From a property-oriented viewpoint, we wish such a translation to preserve logical validities; and the translation of the valid sentences $\forall x(x \approx x)$, $\forall x,x'[x \approx x' \rightarrow x' \approx x)]$ and $\forall x,x',x''[(x \approx x' \wedge x' \approx x'') \rightarrow x \approx x'']$ express the equivalence requirement. Also, the translation of valid equality axioms express the above congruence requirement with respect to the translations of operation and predicate symbols.

We are thus led to considering restrictions on such translations of equality in the same spirit as before. We consider such an interpretation $I:Trv(L') \rightarrow P''$ with translation of equality just in case the equivalence and congruence requirements are consequences of P". With these requirements we achieve the desired goals. But, as before, such interpretations with translation of equality can be reduced to symbol interpretations into quotient sorts with conversions (see 3.8.6); a point we shall return to when considering many-sorted interpretations in 4.9.2.

REFERENCES

Arbib, M. and Mannes, E. (1975) *Arrows, Structures and Functors : the Categorical Imperative*. Academic Press, New York.

Barwise, J. ed. (1977) *Handbook of Mathematical Logic*. North-Holland, Amsterdam.

Bauer, F., L. and Wössner, H. (1982) *Algorithmic Language and Program Development*. Springer-Verlag, Berlin.

Broy, M. (1983) Program construction by transformations: a family of

sorting programs. In Breuman, A. W. and Guiho, G. (eds) *Automatic Program Constructiom*, Reidel, Dordrecht.

Broy, M., Pair, C. and Wirsing, M (1981) A systematic study of models of abstract data types. Centre de Recherche en Informatique de Nancy Res. Rept. 81-R-042, Namcy.

Broy, M. and Pepper, P. (1981) Program development as a formal activity. *IEEE Trans. Software Engin.*; **SE-7** (1)14-22.

Broy, M. and Wirsing, M (1982) Partial abstract data types. *Acta Informatica*, **18** 47-64.

Byers, P. and Pitt, D. (1990) Conservative extensions: a cautionary note. *Bull. EATCS*,41, 196-201.

Chang, C. C. and Keisler, H. J. (1973) *Model Theory*. North Holland, Amsterdam.

Dahl, O., Dijkstra, E. and Hoare, C. (1972) *Structured Programming*. Academic Press, New York.

Darlington, J. (1978) A synthesis of several sorting algorithms. *Acta Informatica*, **11** (1), 1-30.

Ebbinghaus, H. D., Flum, J. and Thomas, W. (1984) *Mathematical Logic*. Springer-Verlag, Berlin.

Enderton, H. B. (1972) *A Mathematical Introduction to Logic*. Academic Press; New York.

Ehrich, H.-D. (1982) On the theory of specification, implementation and parameterization of abstract data types. *J. ACM*, **29** (1), 206-227.

Ehrig, H. and Mahr, B. (1985) *Fundamentals of Algebraic Specifications, 1: Equations and Initial Semantics*. Springer-Verlag, Berlin.

Gehani, N. and McGettrick, A., D. (1986) *Software Specifications Techniques*. Addison-Wesley, Reading.

Ghezzi, C., Jazayeri, M. (1982) *Programming Languages Concepts*. Wiley, New York.

Goguen, J. A.; Thatcher, J. W. and Wagner, E. G. (1978) An initial algebra approach to the specification, correctness and implementation of abstract data types. In Yeh, R. T. (ed.) *Current Trends in Programming Methodology*: Prentice Hall, Englewood Cliffs, . 81-149.

Guttag, J. V (1977) Abstract data types and the development of data structures. *Comm. Assoc. Comput. Mach.*, **20** (6), 396-404.

Guttag, J. V (1980) Notes on type abstraction. *IEEE Trans. Software Engin.*, **6** (1),.

Guttag, J. V. and Horning, J. J. (1978) The algebraic specification of abstract data types. *Acta Informatica*, **10** (1), p. 27 - 52.

Hoare, C. A. R. (1972) Proof of correctness of data representations. *Acta Informatica*, **4**, 271-281.

Hoare, C. A. R. (1974) Notes on data structuring. In Dahl *et al.* 1(974); 83-174.

Hoare, C. A. R. (1978) Data Structures. In Yeh, R. (ed.) *Current Trends in Programming Methodology, Vol IV*. Prentice Hall, Englewood Cliffs, 1-11.

Jackson, M., A. (1980) *Principles of Program Design*. Academic Press, London.

Ledgard, H. and Taylor, R. W. (1977) Two views on data abstraction. *Comm. Assoc. Comput. Mach.*, **20** (6), 382-384.

Maibaum, T. S. E. (1986) The role of abstraction in program development. In Kugler, H.-J. ed. *Information Processing '86*. North-Holland, Amsterdam, 135-142.

Maibaum, T. S. E., Sadler, M. R. and Veloso, P. A. S. (1984) Logical specification and implementation. In Joseph, M. and Shyamasundar R. eds. *Foundations of Software Technology and Theoretical Computer Science*. Springer-Verlag, Berlin, 13-30.

Maibaum, T. S. E. and Turski, W. M. (1984) On what exactly is going on when software is developed step-by-step. *tProc. 7h Intern. Conf. on Software Engin.* IEEE Computer Society, Los Angeles, 528-533.

Maibaum, T. S. E, Veloso, P. A. S. and Sadler, M. R. (1985) A theory of abstract data types for program development: bridging the gap?. In Ehrig, H., Floyd, C., Nivat, M. and Thatcher, J. eds. *Formal Methods and Software Development; vol. 2: Colloquium on Software Engineering*. Springer-Verlag, Berlin, 214-230.

Maibaum, T. S. E, Veloso, P. A. S. and Sadler, M. R. (1991) A logical approach to specification and implementation of abstract data types. Imperial College of Science, Technology and Medicine, Dept. of Computing Res. Rept. DoC 91/47, London.

Manna, Z. (1974) *The Mathematical Theory of Computation*. McGraw-Hill, New York.

Meré, M. C. ; Veloso, P. A. S. (1992) On extensions by sorts.. PUC - RJ, Dept. Informática, Res. Rept. MCC 38/92, Rio de Janeiro..

Pair, C. (1980) Sur les modèles des types abstraites algébriques. Centre de Recherche en Informatique de Nancy Res. Rept. 80-p-042, Namcy.

Parnas, D. L. (1979) Designing software for ease of extension and contraction. *IEEE Trans. Software Engin.*, **5** (2), 128-138.

Pequeno, T. H. C. and Veloso, P. A. S. (1978) Do not write more axioms than you have to. *Proc. Intern. Computing Symposium*, Taipei, 487-498.

Shoenfield, J. R. (1967) *Mathematical Logic*. Addison-Wesley, Reading.

Smirnov, V. A. (1986) Logical relations between theories. *Synthese*, **66**, p. 71 - 87.

Smith, D. R. (1985) The Design of Divide and Conquer Algorithms. *Science Computer Programming*, **5** 37-58.

Smith, D. R. (1990) Algorithm theories and design tactics. *Science of Computer Programming.*, **14**, 305-321.

Smith, D. R. (1992) Constructing specification morphisms. Kestrel Institute, Tech. Rept. KES.U.92.1, Palo Alto.

Turski, W. M and Maibaum, T. S. E. (1987) *The Specification of Computer Programs*. Addison-Wesley, Wokingham.

van Dalen, D. (1989) *Logic and Structure* (2nd edn, 3rd prt). Springer-Verlag, Berlin.

Veloso, P. A. S. (1984) Outlines of a mathematical theory of general problems. *Philosophia Naturalis*, **21** (2/4), 354-362.

Veloso, P. A. S. (1985) On abstraction in programming and problem solving. *2nd Intern. Conf. on Systems Research, Informatics and Cybernetics*. Baden-Baden.

Veloso, P. A. S. (1987) *Verificação e Estruturação de Programas com Tipos de Dados*. Edgard Blücher, São Paulo.

Veloso, P. A. S. (1987) On the concepts of problem and problem-solving method. *Decision Support Systems*, **3** (2), 133-139.

Veloso, P. A. S. (1988) Problem solving by interpretation of theories. In Carnielli, W. A. ; Alcântara, L. P. eds. *Methods and Applications of Mathematical Logic*. American Mathematical Society, Providence, 241-250.

Veloso, P. A. S. (1991) A computing-like example of conservative, non-expansive, extension. Imperial College of Science, Technology and Medicine, Dept. of Computing, Res. Rept. DoC 91/36, London.

Veloso, P. A. S. (1992) Yet another cautionary note on conservative extensions: a simple example with a computing flavour. *Bull. EATCS*, 46, 188-192.

Veloso, P. A. S. (1992) On the modularisation theorem for logical specifications: its role and proof. PUC - RJ, Dept. Informática Res. Rept. MCC 17/92, Rio de Janeiro.

Veloso, P. A. S. (1992) Notes on interpretations of logical specifications. COPPE-UFRJ Res. Rept. ES-277/93, Rio de Janeiro.

Veloso, P. A. S. (1993) The Modularization Theorem for unsorted and many-sorted specifications. COPPE-UFRJ Res. Rept. ES-284/93, Rio de Janeiro.

Veloso, P. A. S. (1993) A new, simpler proof of the Modularisation Theorem for logical specifications. *Bulletin of the IGPL* **1** (1), 1-11.

Veloso. P. A. S. and Maibaum, T. S. E. (1984) What is wrong with errors: incomplete specifications for abstract data types. UFF, ILTC, Res. Rept., Niterói.

Veloso, P. A. S. and Maibaum, T. S. E. (1992) On the Modularisation Theorem for logical specifications. Imperial College of Science, Technology & Medicine, Dept. of Computing Res. Rept. DoC 92/35, London.

Veloso, P. A. S., Maibaum, T. S. E. and Sadler, M. R. (1985) Program development and theory manipulation. In *Proc. 3rd Intern. Workshop on Software Specification and Design*. IEEE Computer Society, Los Angeles, 228-232.

Veloso. P. A. S. and Pequeno , T. H. C. (1978) Interpretations between many-sorted theories. 2nd Brazilian. Colloquium on Logic; Campinas.

Veloso, P. A. S. and Veloso. S. R. M. (1981) Problem decomposition and reduction: applicability, soundness, completeness. In Trappl, R.; Klir, J. ; Pichler, F. eds. *Progress in Cybernetics and Systems Research.* Hemisphere, Washington, DC, 199-203.

Veloso, P. A. S. and Veloso. S. R. M. (1990) On extensions by function symbols: conservativeness and comparison. COPPE-UFRJ Res. Rept. ES-288/90, Rio de Janeiro.

Veloso, P. A. S. and Veloso, S. R. M. (1991) Some remarks on conservative extensions: a Socratic dialogue. *Bull. EATCS*, 43, 189-198.

Veloso, P. A. S. and Veloso, S. R. M. (1991) On conservative and expansive extensions. *O que no faz pensar: Cadernos de Filosofia*, 4, 87, 106.

Veloso, P. A. S. and Veloso, S. R. M. (1991) On conservative and expansive extensions: why and how they differ. Imperial College of Science, Technology & Medicine, Dept. of Computing Res. Rept. DoC 91/30, London.

Wirsing, M., Pepper, P. and Broy, M. (1983) On hierarchies of abstract data types. *Acta Informatica* **20** (1) 1-33.