PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 34/95

Logical Specifications: 4.B - Interpretations of Many-Sorted 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 34/95

Editor: Carlos J. P. Lucena October, 1995

Logical Specifications: 4.B - Interpretations of Many-Sorted 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.B INTERPRETATIONS OF MANY-SORTED 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 34/94

Abstract. We present basic concepts and results concerning interpretations of many-sorted logical specifications, as well as some connections with extensions. We start with simple 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, an internalisation technique via translation diagram, and the many-sorted Modularisation Construction. We next consider the Modularisation Theorem for many-sorted specifications, first some simple variants, then the case of extensions, and next modularity of many-sorted interpretations. Finally, we consider some variants of interpretations found in the literature: translation with relativisation (and its reduction to subsort), translation of equality (and its reduction to quotient), translation with sort tupling (and its reduction to product), as well as general interpretations incorporating these three features, and the reduction of many-sorted logic to unsorted logic via relativisation. Some examples illustrating these ideas are presented. This report is the second half of a draft of the fourth section of a handbook chapter. Other reports cover the remaining sections.

Key words: Formal specifications, many-sorted axiomatic specifications, interpretations of specifications, many-sorted language translations, internalisation of translation, translation diagram, conservative extensions, modularity, joint consistency, translations with relativisation, translation of equality, translations with sort tupling, sort introducing constructs.

Resumo. Apresentamos conceitos e resultados básicos sobre interpretações de especificações lógicas poli-sortidas, bem como algumas conexões com extensões. Começamos com interpretações simples 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 de internalização através de diagrama de tradução e a versão poli-sortida da Construção de Modularização. Tratamos então do Teorema da Modularização para especificações poli-sortidas, começando com algumas variantes simples e o caso de extensões, para então passar a modularidade de interpretações poli-sortidas. Por fim, consideramos algumas variantes de interpretações encontradas na literatura: tradução com relativização (e sua redução a subsorte), tradução da igualdade (e sua redução a quociente), tradução com tuplagem (e sua redução a produto), bem como interpretações generalizadas, incorporando estas três características, e a redução da lógica poli-sortida à lógica sem sortes através de relativização. Apresentam-se também alguns exemplos que ilustram essas idéias. Este relatório é a segunda metade de um esboço da quarta seção de um capítulo de um manual. Outros relatórios cobrem as demais seções.

Palavras chave: Especificações formais, especificações axiomáticas poli-sortidas, interpretações de especificações, traduções de linguagens poli-sortidas, internalização de tradução, diagrama de tradução, extensões conservativas, modularidade, consistência conjunta, traduções com relativização, tradução da igualdade, traduções com tuplagem de sortes, construtores de sortes.

NOTE

This report is the second half 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 preceding report in this series was Logical Specifications: 4.A Interpretations of unsorted Specifications, covering the unsorted version of the present one, namely:

unsorted translations and interpretations;
composition, decomposition and internalisations of translations;
tions of translations and interpretations, relativisation, translation
equality, sort tupling;
interpolation 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. [illegible], Haydée W. Poubel, M. Claudia Meré, Tarcísio H. C. Pequeno and Roberto Lins de Carvalho

CONTENTS [1]

[1] See the preceding note for an explanation of the numbering system.

[2] The report Logical Specifications: 4.A Interpretations of unsorted Specifications covers the unsorted version of the present one.

List of Figures

4.6 Many-sorted interpretations [1]

We shall now examine the extensions of the previous concepts and results[2] to the many-sorted case: translations of many-sorted languages and interpretations of many-sorted specifications.

These extensions are generally easy, but the notation is somewhat more elaborate. The basic ideas are, as before, translating formulae to formulae and structures to stuctures in a property preserving manner. We start with the simple case of symbol to symbol translation, and later consider some possible variations in 4.9.

4.6.1 Simple many-sorted translations

We begin by considering simple translations of languages, which translate symbols to symbols, enabling syntactical translations of properties expressed in the language and reverse semantical translations of structures, as before.

A. *Syntactic many-sorted translation*

We shall now consider a many-sorted language to have a single *equality symbol* $\approx$, instead of an *equality symbol* $\approx_s$ for each sort s. Thus, the equality formulae are now of the form $t \approx t'$, where t and t' are terms of the same sort, say s (instead of the previous $t \approx_s t'$).

A (simple) translation from a language L_1 to language L_2 is a mapping that assigns to each symbol of the former a symbol of the latter, respecting their declarations. Now, that sorts are translated, their variables are translated as well.

More precisely, consider many-sorted languages L_i with sets of sorts $Srt(L_i)=S_i$, of predicate symbols $Prd(L_i)=R_i$, of operation symbols $Opr(L_i)=F_i$, and of variables $Var(L_i)=V_i$, i=1,2. A (*simple*) *translation* (or *renaming*) I from *source language* L_1 to *target language* L_2 - denoted $I:L_1 \rightarrow L_2$ - is a mapping, consisting of renaming of sorts, variables, operation and predicate symbols, respecting syntactical declarations, in the following sense.

- Sort renaming $I_S:S_1 \rightarrow S_2$ assigns to each sort $s \in S_1$ a sort $I_S(s) \in S_2$.
- Predicate renaming $I_R:R_1 \rightarrow R_2$ assigns to each predicate symbol $r \in R_1$, over sorts $s_1,\ldots,s_m$ in L_1, a predicate symbol $I_R(r) \in R_2$, over sorts $I_S(s_1),\ldots,I_S(s_m)$ in L_2.
- Operation renaming $I_F:F_1 \rightarrow F_2$ assigns to each operation symbol $f \in F_1$,

---

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

[2] The unsorted case is presented in the report Logical Specifications: 4.A. Interpretations of unsorted Specifications.

from sorts $s_1,\ldots,s_n$ to s in L_1, an operation symbol $I_F(f)\in F_2=Opr(L_2)$, from sorts $I_S(s_1),\ldots,I_S(s_n)$ to $I_S(s)$ in L_2.

- Variable renaming $I_V: V_1 \rightarrow V_2$ assigns to each variable $v\in V_1$, ranging over sort s in L_1, a variable $I_V(v)\in V_2$, ranging over sort $I_S(s)$ in L_2.

We shall generally denote each one of these mappings, or their (disjoint) union, simply by I.

We can regard a many-sorted language L with sets S of sorts, R of predicate symbols, F of operation symbols, and V of variables, as an S-sorted set with declaration d assigning:

- to each variable $v\in V$ the sort $d(v)\in S$ over which it ranges;
- to each predicate symbol $r\in R$ its profile $d(r)=<s_1\ldots s_m>\in S^*$ of sorts;
- to each operation symbol $f\in F$ its profile $d(f)=<s_1\ldots s_n,s>\in S^*$ of sorts.

Now, a sort renaming $I:S_1\rightarrow S_2$ extends naturally to a renaming of sequences of sorts $I^*: S_1{}^*\rightarrow S_2{}^*$. So, the above requirement of 'respecting syntactical declarations' becomes the requirement for I to be a language homomorphism, making the diagrams in figure 4.14 commute.

$$\begin{array}{ccc} V_1 & \xrightarrow{I_V} & V_2 \\ {\scriptstyle d_1}\downarrow & & \downarrow{\scriptstyle d_2} \\ S_1 & \xrightarrow{I_S} & S_2 \end{array} \qquad \begin{array}{ccc} R_1 & \xrightarrow{I_R} & R_2 \\ {\scriptstyle d_1}\downarrow & & \downarrow{\scriptstyle d_2} \\ S_1^* & \xrightarrow{I_S^*} & S_2^* \end{array} \qquad \begin{array}{ccc} F_1 & \xrightarrow{I_F} & F_2 \\ {\scriptstyle d_1}\downarrow & & \downarrow{\scriptstyle d_2} \\ S_1^* & \xrightarrow{I_S^*} & S_2^* \end{array}$$

Fig. 4.14: Translation preserving declarations

As in the unsorted case, the translation I of symbols of L_1 to L_2 can be naturally extended to translate terms and formulae. The *translation of a term* $t\in Trm(L_1)$ via I is the term $I(t)\in Trm(L_2)$ obtained by replacing each occurrence of a symbol of L_1 by its translation according to the mapping I. Notice that this translation preserves sorts of terms: it translates term $t\in Trm(L_1)[s]$, of sort s in L_1,to term $I(t)\in Trm(L_2)[I(s)]$, of sort s in L_2. The *translation of a formula* $\varphi\in Frml(L_1)$ via I is the formula $I(\varphi)\in Frml(L_2)$ - obtained by replacing each occurrence of a symbol of L_1 by its translation according to the mapping I. Each quantification part $(\forall u{:}s)$, or $(\exists u{:}s)$, is replaced by $(\forall w{:}t)$, respectively $(\exists w{:}t)$, where $I_V(u)=w$ and $I_S(s)=t$; and each equality formula $t\approx t'$ of L_1, with terms $t,t'\in Trm(L_1)[s]$, is replaced by the equality $I(t)\approx I(t')$ of L_2.

Notice that these mappings are not necessarily surjective. Neither are they required to be injective; except for the renaming of variables. For reasons mentioned in 4.5.3, we do not wish two distinct variables to be mapped to the same target variable. We require this injectivity even for variables ranging over distinct source sorts. For, imagine that source

variables $u_1{:}s_1$ and $v_2{:}s_2$ are mapped to the same target variable w:t. This has the following unpleasant consequences.

- The non-contradictory sentence $(\exists x', u_1{:}s_1)(\forall y'{:}s_2)(\exists v_2{:}s_2)[\neg x' \approx u_1 \wedge v_2 \approx y']$ is translated to $(\exists x'', w{:}t)(\forall y''{:}t)(\exists w{:}t)[\neg x'' \approx w \wedge w \approx y'']$, where $I(x')=x''$ and $I(y')=y''$, and the latter entails the contradictory $(\exists x''{:}t)(\forall y''{:}t)[\neg x'' \approx y'']$.
- For a unary source operation symbol f' from sort s_1 to s_2, the sentence $(\exists u_1{:}s_1)(\exists v_2{:}s_2) f'(u_1) \approx v_2$ is logically valid, which does not happen with its translation $(\exists w{:}t) f''(w) \approx w$, which asserts that the realisation of $f''=I(f')$ has a fixed point.
- For a binary source predicate symbol r' over sorts s_1 and s_2 the sentence $(\exists u_1{:}s_1)(\exists v_2{:}s_2) r'(u_1, v_2)$ asserts that its realisations are nonempty. But, its translation $(\exists w{:}t) r''(w,w)$ asserts the much stronger requirement that the realisation of $r''=I(r')$ meets the identity.

B. *Semantical many-sorted translation*

As in the unsorted case, a simple translation of many-sorted languages induces a connection between structures (in the 'reverse' direction), which permits discussing structures for one language in the other.

Let $I{:}L_1 \rightarrow L_2$ be a (simple) translation from source language L_1 to target language L_2. By composing the assignments given by the translation and the structure for L_2, we obtain realisations for the symbols of L_1 (see figure 4.1). Given a structure $\mathfrak{M}$ for target language L_2, the *structure induced by* $\mathfrak{M}$" under I is the structure $\mathfrak{M}\downarrow I$ for source language L_1, where

its universe of sort $s \in S_1$ is the universe of $I(s) \in S_2$ in $\mathfrak{M}$: $\mathfrak{M}\downarrow I[s] = \mathfrak{M}[I(s)]$;

- for each predicate symbol $r \in R_1$, its realisation in the induced structure $\mathfrak{M}\downarrow I$ is the realisation of its translation $I(r) \in R_2$ in the given structure $\mathfrak{M}$: $\mathfrak{M}\downarrow I[r] = \mathfrak{M}[I(r)]$;
- for each operation symbol $f \in F_1$, its realisation in the induced structure $\mathfrak{M}\downarrow I$ is the realisation of its translation $I(f) \in F_2$ in the given structure $\mathfrak{M}$: $\mathfrak{M}\downarrow I[f] = \mathfrak{M}[I(f)]$.

Now, an assignment a of values in structure $\mathfrak{M}$ to the variables of L_2 assigns to each variable $v \in Var(L_2)$, ranging over sort $s \in S_2$ of L_2, a value $a(v) \in M[s]$ in the universe of sort s in $\mathfrak{M}$. Thus, by composing a translation $I{:}L_1 \rightarrow L_2$ with this assignment we obtain an assignment I;a of values in the induced structure $\mathfrak{M}\downarrow I$ to the variables of L_1. We thus have a Translation Connection, as in the unsorted case.

Proposition Translation Connection (many-sorted case)

Let $I{:}L_1 \rightarrow L_2$ be a (simple) translation from source language L_1 to target

language L_2. Consider a structure $\mathfrak{M}$ for L_2 with induced structure $\mathfrak{M}\downarrow I$. Then, for each formula φ of source language L_1, we have

a) for every assignment a of values in M to the variables of L_2 $\mathfrak{M}\downarrow I \models \varphi$ [I;a] iff $\mathfrak{M} \models I(\varphi)$ [a];

b) the realisation of formula $\varphi \in Frml(L_1)$ in the induced structure and the realisation of its translation in the original structure are the same: $\mathfrak{M}\downarrow I[\varphi] = \mathfrak{M}[I(\varphi)]$.

The proof of this result is, as in the unsorted case, by structural induction. We just mention that the injectiveness requirement for renaming of variables is important for the cases of quantification.

4.6.2 Simple many-sorted interpretations

We shall now consider interpretations of many-sorted specifications by adapting the ideas of the unsorted versions in 4.2.3.

Consider specifications $P_1 = <L_1, G_1>$ and $P_2 = <L_2, G_2>$. A *simple interpretation* from *source specification* P_1 to *target specification* P_2 - denoted $I: P_1 \rightarrow P_2$ - is a language translation $I: L_1 \rightarrow L_2$ preserving logical consequences, in the following sense: for every sentence σ of L_1 if $P_1 \models \sigma$, then $P_2 \models I(\sigma)$, i. e. I translates the source theory $Cn[P_1]$ into the target theory $Cn[P_2]$. As in the unsorted case, a *faithful* interpretation is one where $P_1 \models \sigma$ iff $P_2 \models I(\sigma)$ for every sentence σ of L_1.

Theorem Interpretation Theorem (for many-sorted specifications)

Consider a language translation $I: L_1 \rightarrow L_2$. Given specifications $P_1 = <L_1, G_1>$ $P_2 = <L_2, G_2>$, the following are equivalent.

a) I is an interpretation from P_1 to P_2.

b) For every axiom $\gamma \in G_1$, its translation $I(\gamma)$ is a consequence of P_2.

c) For every structure $\mathfrak{M} \in Mod[P_2]$, the induced structure $\mathfrak{M}\downarrow I$ is a model of P_1.

The proposition characterising those interpretations which are faithful also carries over to the many-sorted case.

4.7. Operations on specifications and interpretations

We shall now extend to the many-sorted case the ideas in 4.3.

4.7.1 Operations on many-sorted specifications

We now extend the ideas in 4.3.1 concerning languages and specifications within the same context to the many-sorted case. It will be seen that the main difference lies in the part concerning languages, because of the presence of sorts.

A. *Union and intersection (many-sorted case)*

When operating with many-sorted languages, we must generally start with the sets of sorts and then proceed to the rest.

As before, we call (many-sorted) languages L' and L" *compatible* iff they are sub-languages of some language L. In this case, there is no confusion of symbols and we can form intersection or union of languages.

Consider a (many-sorted) language L with set of sorts S, alphabet A, set of variables V, and declaration d. A *sub-signature of* L consists of subsets $T\subseteq S$ and $B\subseteq A$, such that, for every $b\in B\subseteq A$, d(b) involves only sorts of the subset T. Now, such a sub-signature of L, consisting of $T\subseteq S$ and $B\subseteq A$, gives rise to a sub-language of L, with set of sorts T, alphabet B, set of variables $V|T:=\{v\in V/d(v)\in T\}$, and the declaration being the appropriate restriction of d; we call such a language the *restriction of* L *to the sub-signature consisting of* $T\subseteq S$ *and* $B\subseteq A$. (In case $T=\emptyset$, a sub-signature of L must have $B=\emptyset$, which leads to the restriction $\emptyset$, with no sorts or variables and empty alphabet. This is an uninteresting limit case, for it has no terms or formulae.)

Consider a family of sub-languages $L_i=<S_i,A_i,V_i>$ of $L=<S,A,V>$, for $i\in I$. Notice that we have a sub-signature of L consisting of $S_\cap:=\cap_{i\in I}S_i$ and $A_\cap:=\cap_{i\in I}A_i$; by the *intersection language* $\cap_{i\in I}L_i$ we mean the restriction of L to this sub-signature. We also have a sub-signature of L consisting of $S^\cup:=\cup_{i\in I}S_i$ and $A^\cup:=\cup_{i\in I}A_i$; by the *union language* $\cup_{i\in I}L_i$ we mean the restriction of L to this sub-signature.

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 sorts: $Srt(L')\cap Srt(L")=\emptyset$.

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

B. *Orthogonal families (many-sorted case)*

The idea of orthogonal families of languages and their properties of joint expansiveness seen in 4.3.1.B extend to the many-sorted case.

An *orthogonal family of languages* is a family of sub-languages L_i of some language L, for $i\in I$, such that all $j\neq k$ in I $L_j\cap L_k\subseteq L_\cap:=\cap_{i\in I}L_i$. Such orthogonal families of languages have the following properties (see 4.3.1.B).

Joint expandability:

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 $\mathfrak{B}_j\downarrow L_\cap=\mathfrak{B}_j\downarrow L_\cap$ to $L_\cap$ has a common expansion $\mathfrak{C}$ to $L^\cup$: $\mathfrak{B}_i\subseteq\mathfrak{C}$ for each $i\in I$.

Orthogonal expansiveness:

Given a family $P_i=<L_i,G_i>$ of extensions of $P=<L_\cap,G>$, for $i\in I$, let $P^\cup:=\cup_{i\in I}P_i$. If $P\preceq P_i$, for every $i\in I$, then $P\preceq P^\cup$.

4.7.2 Decomposition of (many-sorted) interpretations

We shall now examine some simple decompositions of (many-sorted) interpretations, similar to those seen in 4.3.2.

The concepts of pre and post images as well as the associated decompositions seen in 4.3.2.B carry over to the many-sorted case without any conceptual difficulty.

Also, the decompositions induced by language partitions of 4.3.2.C extend to many-sorted versions with due consideraton of the fact that sorts must be treated before the symbols touching them (see the remarks concerning symbol removal in 3.4 and 3.8).

The preceding remark suggests a decomposition of many-sorted interpretations based on sorts (see fig 4.15). Consider a (simple) translation $I:L'\rightarrow L"$ from $L'=<S',A',V'>$ to $L"=<S",A",V">$. We will construct an intermediate language $L°=<S°,A°,V°>$ and translations $I^*:L'\rightarrow L°$ and $I^\#:L°\rightarrow L"$ so that I can be decomposed as $I^*;I^\#$.

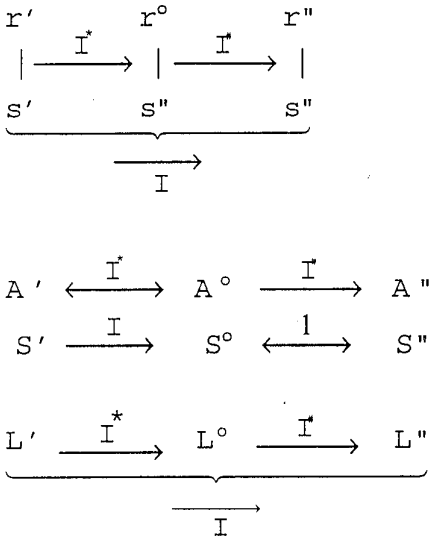

Fig. 4.15: Sort based decomposition of translation .

We first construct language $L°=<S°,A°,V°>$ by taking $S°:=S"$, $V°:=V"$ and defining $A°$ as follows: for each symbol $a'\in A'$, $A°$ has a corresponding symbol $a°$ whose profile is obtained from the profile of a' by replacing each sort s by its translation I(s). We now define the translations:

- translation I^* acts as I over S' and V'; over A', I^* maps (bijectively) each symbol $a'\in A'$ to its corresponding one $a°\in A°$;

- translation $I^{\#}$ is the identity on $S^{\circ}=S''$ and $V^{\circ}=V''$; over A°, $I^{\#}$ maps each symbol $a^{\circ} \in A^{\circ}$ to $I(a') \in A''$ where $I^{*}(a')=a^{\circ}$.

For instance, consider formula r'(v') of L', with v':s', which I translates to r''(v''), with v'':s'' in L''. Then, I^{*} translates r'(v') to $r^{\circ}(v'')$, where r° is a unary predicate symbol over sort s'' in L°; and $I^{\#}$ translates $r^{\circ}(v'')$ to r''(v'') in L''.

As before, if I interprets $P=\langle L,G\rangle$ into $P'=\langle L',G'\rangle$, the above decomposition can be used to obtain a specification $P^{\circ}=\langle L^{\circ},G^{\circ}\rangle$ so that $I^{*}:P' \rightarrow P^{\circ}$ and $I^{\#}:P^{\circ} \rightarrow P''$.

4.7.3 Internalisation via translation diagrams

The notion of induced structure shows how a structure for the target language defines - via the translation - a structure for the source language. We shall now indicate how this idea can be 'internalised' (Veloso 1993). We can 'internalise' the translation by coding its information into sentences of an appropriately richer language - one expressing that each source symbol is equivalent to its translation. This set of sentences will describe a process of structure induction.

Consider first the unsorted case, to get the gist of the idea. Given a translation $I:L' \rightarrow L''$, we may, by resorting to renaming if necessary, assume that the source and target languages L' and L'' are disjoint and compatible. We then construct the union language $L' \cup L''$, as in 4.3.1, which has alphabet $A' \cup A''$, and common set of variables.

We can now 'internalise' the translation by coding its information into a set $\Phi[I]$ of formulae $a \leftrightarrow I(a)$ of $L' \cup L''$ - expressing the fact that each source symbol is equivalent to its translation. More precisely, $\Phi[I]$ consists of the following sentences of $L' \cup L''$:

- $\forall v_1 \ldots \forall v_m [r'(v_1,\ldots,v_m) \leftrightarrow r''(v_1,\ldots,v_m)]$, for each m-ary predicate symbol r' of L', with I(r')=r''; and
- $\forall x_1 \ldots \forall x_n [f'(x_1,\ldots,x_n) \approx f''(x_1,\ldots,x_n)]$, for each n-ary operation symbol f' of L', with I(f')=f''.

Then, it is not difficut to see that we have a description of the process of structure induction: every structure $\mathfrak{M}$ for the union language $L' \cup L''$ has reducts $M\mathfrak{M}\downarrow L'$ and $\mathfrak{M}\downarrow L''$; and $\mathfrak{M} \models \Phi[I]$ iff the reduct $\mathfrak{M}\downarrow L'$ is the structure $(\mathfrak{M}\downarrow L'')\downarrow I$ induced by the reduct $\mathfrak{M}\downarrow L''$. Notice that $\Phi[I]$ provides the information about the translation in the form of an extension of $\langle L'',\varnothing\rangle$ by definitions of the symbols of L', a remark which can be used for an alternative proof for the Translation Connection.

For a translation $I:L' \rightarrow L''$, its (internalised) kernel $Krnl[I]=\langle L',\Lambda[I]\rangle$ (see 4.3.3) provides part of the information in its diagram $Dgrm[I]=\langle L'+L'',\Phi[I]\rangle$,

namely which symbols have the same translation, but not the translations themselves. This connection can be expressed more precisely. Consider the extension of translation $I:L' \to L''$ by the identity (since $L' \cap L'' = \emptyset$) to $I^{\dagger}:L' \cup L'' \to L''$. We then have the equivalence $Dgrm[I] \equiv Krnl[I^{\dagger}]$, as well as an expansive extension $Krnl[I] = \langle L', \Lambda[I] \rangle \preceq \langle L' \cup L'', \Phi[I] \rangle = Dgrm[I]$.

In the many-sorted case, we must construct a language that is appropriate for expressing the equivalence between a source symbol and its translation by connecting source and target sorts. The idea is that, in the presence of bijective connections, a source symbol and its translation become interdefinable.

Consider a translation $I:L' \to L''$ from language $L' = \langle S', A', V' \rangle$ to language $L'' = \langle S'', A'', V'' \rangle$. We may, by resorting to renaming if necessary, assume that L' and L'' are disjoint sub-languages (of some sufficiently large language L). We then form the union language $L' \cup L''$, which has sets of sorts $S' \cup S''$ and of variables $V' \cup V''$, and alphabet $A' \cup A''$.

$$s \xrightarrow{\ I\ } t \Rightarrow \underbrace{s \xrightarrow{\ j_s\ } t}_{\beta[I](s)}$$

Fig. 4.16: Sort connection sentence $\beta[I](s)$

We first connect the sorts in S' and S'' via I_S. We select (from L) a set J[I] of new symbols (not in L' or L'') for unary functions: a (conversion) ation symbol j_s with source sort $s \in S'$ and target sort $I(s) \in S''$, for each s of L'. We thus have a *sort connection language* K[I] with set of sorts $S' \cup S''$, alphabet $J[I] = \{j_s(s) \to t / s \in S'\}$, and set of variables $V' \cup V''$. Now, for each source sort, we have a sentence of K[I] expressing that each connection is a bijection. More precisely, for each sort s of L', with I(s)=t, its *sort connection sentence* $\beta[I](s)$ is $(\forall y{:}t)(\exists !x{:}s) j_s(x) \approx y$. We now collect all these sentences to obtain the *sort connection diagram* $SrCnt[I] := \langle K[I], \Sigma[I] \rangle$ of I, where $\Sigma[I] := \{\beta[I](s) / s \in S'\}$. The idea of SrCnt[I] is that a sort $s \in S'$ and its translation $I(s) \in S''$ are interexplicable via $\beta[I](s)$ (see figure 4.16).

$$L' \left\{ \begin{array}{ccc} A' & & A'' \\ & \underbrace{S' \xrightarrow{\ J[I]\ } S''}_{K[I]} & \end{array} \right\} L''$$

Fig. 4.17: Structure of diagram language $L[I] = L' \cup K[I] \cup L''$

We now extend the sort connection language K[I] by adding the extra-logical symbols in the union alphabet $A' \cup A''$, to obtain the *alphabet matching language* L[I] with set of sorts $S' \cup S''$, alphabet $A' \cup J[I] \cup A''$, and set

of variables V'∪V". The structure of these languages - L'=<S',A'>, L"=<S",A">, K[I]=<S'∪S",J[I]>, and L[I]=<S'∪S",A'∪J[I]∪A"> - is depicted in figure 4.17.

Now, for each source symbol, we have a sentence of L[I] expressing that it has the same behaviour as its translation. More precisely, for each symbol a of source alphabet A', we construct its *alphabet matching sentence* μ[I](a) in L[I], as follows:

- for each m-ary predicate symbol p, over sorts $s_1,\dots,s_m$ in L', with $I(s_1)=t_1,\dots,I(s_m)=t_m$ and I(p)=q, μ[I](p) is the universal closure of $p(x_1,\dots,x_m)\leftrightarrow q(j_{s_1}(x_1),\dots,j_{s_m}(x_m))$ (see figure 4.18);
- for each n-ary operation symbol f, from sorts $s_1,\dots,s_n$ to s in L', with $I(s_1)=t_1,\dots,I(s_n)=t_n$, I(s)=t and I(f)=g, μ[I](f) is the universal closure of $f(x_1,\dots,x_n)\approx x\leftrightarrow g(j_{s_1}(x_1),\dots,j_{s_n}(x_n))\approx j_s(x)$ (see figure 4.19).

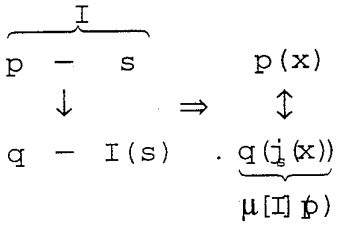

Fig. 4.18: Matching sentence μ[I](p) for (unary) predicate

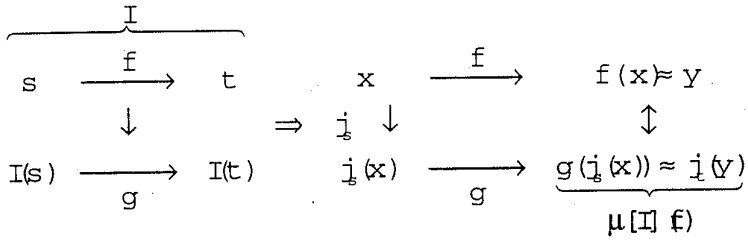

Fig. 4.19: Matching sentence μ[I](f) for (unary) operation

We now collect all these sentences to obtain a specification: the *alphabet matching diagram* AlMch[I]:=<L[I],Δ[I]> of I, where Δ[I]:={μ[I](a)/a∈ A'}. Finally, by the *(full) diagram of translation* I:L'→ L" we mean the specification Dgrm[I]:=<L[I],Φ[I]>, where Φ[I]:=Σ[I]∪Δ[I]. The idea of Dgrm[I] is that, in the presence of Σ[I], a symbol a∈ A' and its translation I(a)∈ A" become interdefinable via μ[I](a).

It is not surprising that the (full) diagram of a translation internalises the information of the translation. In particular, it internalises the process of structure induction. This is the content of the next result, which gives a model-oriented characterisation of the diagram of translation.

Lemma Models of translation diagrams

Consider a translation I:L'→L" with translation diagrams SrCnt[I]:=<K[I],Σ[I]> and Dgrm[I]=<L[I],Φ[I]>.

a) Given a structure $\mathfrak{K}$ for language K[I], $\mathfrak{K}\models\Sigma[I]$ iff, for each s∈S', with I(s)=t∈S", $\mathfrak{K}[j_S]:\mathfrak{K}[s]\rightarrow\mathfrak{K}[t]$ is a bijection from $\mathfrak{K}[s]$ onto $\mathfrak{K}[t]$.

b) Given a structure $\mathfrak{D}$ for language L[I], $\mathfrak{D}\models\Phi[I]$ iff the family $\mathfrak{D}\downarrow K[I]$ of functions $\mathfrak{D}[j_S]:\mathfrak{D}[s]\rightarrow\mathfrak{D}[I(s)]$, for s∈S', is an isomorphism from the reduct $\mathfrak{D}\downarrow L'$ onto the structure $(\mathfrak{D}\downarrow L")\downarrow I$ induced by the reduct $\mathfrak{D}\downarrow L"$.

Proof.

a) By construction (see figure 4.16),

$\mathfrak{K}\models\beta[I](s)$ iff function $\mathfrak{K}[j_S]:\mathfrak{K}[s]\rightarrow\mathfrak{K}[t]$ is a bijection from $\mathfrak{K}[s]$ onto $\mathfrak{K}[t]$.

b) By the definition of reduct and induced structure, we have $(\mathfrak{D}\downarrow L")\downarrow I)[l]=(\mathfrak{D}\downarrow L")[I(l)]=\mathfrak{D}[I(l)]$, for each l∈S'∪A'.

", consider the reduct $\mathfrak{K}=\mathfrak{D}\downarrow K[I]$ of $\mathfrak{D}$ to K[I].

By the construction of alphabet matching sentences, we have:

- for a predicate symbol p, over sorts $s_1,\ldots,s_m$ in L', (see figure 4.18)
  $\mathfrak{D}\models\mu[I](p)$ iff $\mathfrak{D}[j_{s_1}],\ldots,\mathfrak{D}[j_{s_m}]$ map $\mathfrak{D}[p]$ homomorphically to $\mathfrak{D}[I(p)]$;
- for an operation symbol f, from sorts $s_1,\ldots,s_n$ to s in L', (see figure 4.19)
  $\mathfrak{D}\models\mu[I](f)$ iff $\mathfrak{D}[j_{s_1}],\ldots,\mathfrak{D}[j_{s_m}]$ and $\mathfrak{D}[j_S]$ map $\mathfrak{D}[f]$ homomorphically to $\mathfrak{D}[I(f)]$.

Thus, the assertion follows from part (a).

QED

The diagram of a translation matches each source symbol to its translation. One can expect it to match a source sentence to its translation, which is the content of the next result, illustrated in figure 4.20.

$$\begin{array}{ccc} \overbrace{<L',G'>}^{P'} & \xrightarrow{\ I\ } & \overbrace{<L',I(G')>}^{I(P')} \\ \cap & & \cap \\ \underbrace{<L[I]G'\cup\Phi[I]>}_{P'\cup Dgrm[I]} & \cong & \underbrace{<L[I]\Phi[I]\cup I(G')>}_{Dgrm[I]\cup I(P')} \end{array}$$

Fig. 4.20: Diagram extension and translation

Proposition Diagram and translation

Consider a translation I:L'→L" with (full) diagram Dgrm[I]=<L[I],Φ[I]>.

a) For every sentence σ of L': $\Phi[I]\models[\sigma\leftrightarrow I(\sigma)]$.

b) Given a specification P'=<L',G'>, with postimage I(P')=<L",I(G')>, consider the diagram extensions P'∪Dgrm[I]=<L[I],G'∪Φ[I]> and Dgrm[I]∪I(P') = <L[I],Φ[I]∪I(G')>. Then we have:

(i) an equivalence P'∪Dgrm[I]≡Dgrm[I]∪I(P'); and

(ii) an extension I(P')⊆P'∪Dgrm[I].

Proof

a) Consider a sentence $\sigma \in$ Sent(L') and a model $\mathfrak{B} \in$ Mod[Dgrm[I]].
By the preceding result characterising the models of Dgrm[I], we have an isomorphism between $\mathfrak{B}\downarrow$L' and $(\mathfrak{B}\downarrow$L")$\downarrow$I. Thus $\mathfrak{B} \models [\sigma \leftrightarrow I(\sigma)]$.
Indeed, we have successively: $\mathfrak{B} \models \sigma$ iff {since $\sigma \in$ Sent(L')} $\mathfrak{B}\downarrow$L' $\models \sigma$ iff {by the isomorphism} $(\mathfrak{B}\downarrow$L")$\downarrow$I $\models \sigma$ iff {by the Translation Connection} $\mathfrak{B}\downarrow$L" $\models I(\sigma)$ iff {since $I(\sigma) \in$ Sent(L")} $\mathfrak{B} \models I(\sigma)$. Hence $\mathfrak{B} \models [\sigma \leftrightarrow I(\sigma)]$.

{ A property-oriented argument can be obtained by using

- $\Phi[I] \models j_{s_1}(x_1) \approx y_1 \wedge \ldots \wedge j_{s_n}(x_n) \approx y_n \wedge j_s(x) \approx y \rightarrow [f(x_1,\ldots,x_n) \approx x \leftrightarrow g(y_1,\ldots,y_n) \approx y]$,
- $\Phi[I] \models j_{s_1}(x_1) \approx y_1 \wedge \ldots \wedge j_{s_m}(x_m) \approx y_m \rightarrow [p(x_1,\ldots,x_m) \leftrightarrow q(y_1,\ldots,y_m)]$;

as a basis for induction on terms and formulae (relying on $\Sigma[I]$), to establish that, if $I(\varphi(x_1,\ldots,x_m))=\psi(y_1,\ldots,y_m)$, then

- $\Phi[I] \models j_{s_1}(x_1) \approx y_1 \wedge \ldots \wedge j_{s_m}(x_m) \approx y_m \rightarrow [\varphi(x_1,\ldots,x_m) \leftrightarrow \psi(y_1,\ldots,y_m)]$.}

b) Both assertions follow immediately form part (a).

i) Indeed, for each axiom $\gamma \in$ G', $\Phi[I] \models [\gamma \leftrightarrow I(\gamma)]$.

ii) We have $I(P') \subseteq Dgrm[I] \cup I(P') \equiv P' \cup Dgrm[I]$ by (i).

QED

The diagram of I:L'→L" provides the information of the translation in the form of a special extension of Trv(L"):=<L",∅>, which is expected to be expansive. This is a corollary of the next result, which uses extensions by translation diagram to characterise interpretations, as in figure 4.21.

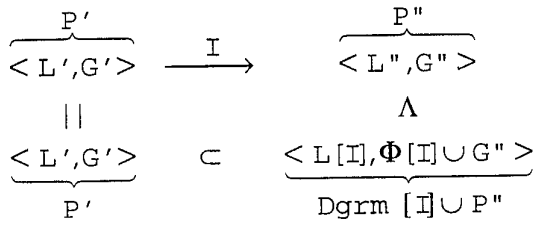

Fig. 4.21: Interpretation characterised by translation diagram

Proposition Characterisation of interpretations by translation diagrams
Consider a translation I:L'→L" with (full) diagram Dgrm[I]=<L[I],Φ[I]>.

Given specifications P'=<L',G'> and P"=<L",G">, consider the diagram extension Dgrm[I]∪P":=<L[I],G"∪Φ[I]>.

a) Dgrm[I]∪P" is an expansive, hence conservative, extension of P".

b) I interprets P' into P" iff Dgrm[I]∪P" extends P'.

c) I interprets P'=<L',G'> faithfully into P" iff Dgrm[I]∪P" is a conservative extension of P'.

Proof (see figure 4.22)

a) We will show P"$\preceq$SrCnt[I]∪P"$\preceq$AlMch[I]∪SrCnt[I]∪P"=Dgrm[I]∪P".

We first show that P"$\preceq$SrCnt[I]∪P" where SrCnt[I]∪P":=<K[I]∪L",Σ[I]∪G">

We can add each sort one at a time. Given a sort s∈ S', consider the extension P_s:=<L_s,{β[I](s)}∪G">, where L_s:=<{s}∪S",{j_s}∪A">.

We have an extension P"$\preceq$Unrlt by definition of unary predicate symbol r_s ?r sort t with defining axiom $(\forall y{:}t)[r_s(y)\leftrightarrow y\approx y]$ (see 3.5.1). We now .ıtroduce s as the subsort of t relativised to relativisation predicate r_s, with insertion $j_s(s)\rightarrow t$; so Unrlt$\preceq$Sbsrt (see 3.8.3).

On the other hand, Sbsrt is equivalent to the extension of P_s by definition of unary predicate symbol r_s over sort t with defining axiom $(\forall y{:}t)[r_s(y)\leftrightarrow(\exists x{:}s)y\approx j_s(x)]$ (see 3.8.3). Thus P"$\preceq P_s$.

Now, for all s≠t in S' $L_s\cap L_t\subseteq$L". So, orthogonal expansiveness (see 4.7.1.B) yields P"$\preceq\cup_{s\in S'}P_s$, and $\cup_{s\in S'}P_s$=SrCnt[I]∪P".

We now show SrCnt[I]∪P"$\preceq$AlMch[I]∪SrCnt[I]∪P".

As already remarked, in the presence of Σ[I], the alphabet matching :ence μ[I](a) is (equivalent to) a proper defining axiom of symbol a∈ A' .n terms of its translation I(a)∈ A". Hence, as above, we have SrCnt[I]∪P"$\preceq${μ[I](a)}∪SrCnt[I]∪P"$\preceq$AlMch[I]∪SrCnt[I]∪P".

{ Alternatively, given $\mathfrak{A}\in$ Mod[P"], we first expand it to $\mathfrak{B}\in$ Mod[SrCnt[I]∪P"], by taking $\mathfrak{B}$[s]=$\mathfrak{A}$[I(s)] and $\mathfrak{B}$[j_s] as the identity function on $\mathfrak{A}$[I(s)]. Then, we expand $\mathfrak{B}\in$ Mod[SrCnt[I]∪P"] to $\mathfrak{C}\in$ Mod[Dgrm[I]∪P"] by using μ[I](a) to define $\mathfrak{C}$[a] in terms of $\mathfrak{B}$[I(a)] for each a∈ A'.}

b) By part (a) and item (a) of the proposition on diagram and translation, for every σ∈ Sent(L'), G"⊨ I(σ) iff Φ[I]∪G"⊨ I(σ) iff Φ[I]∪G"⊨ σ.

Thus, I(Cn[P'])$\subseteq$Cn[P"] iff Cn[P']$\subseteq$Cn[Dgrm[I]∪P"].

c) Similarly to (b), Γ^1(Cn[P"])$\subseteq$Cn[P'] iff Cn[Dgrm[I]∪P"]$\subseteq$Cn[P'].

QED

These properties of translation diagrams enable one to reduce, to a large extent, interpretations to extensions. They will be applied to reduce modularity of many-sorted interpretations to that of extensions.

$$
\begin{array}{rcl}
P'' & = & \overbrace{<S'' , A'', G''>}^{L''} \\
\bigwedge & & \\
\{\beta[I](s)\}P'' & = & <\{s\}\cup S'' , \{j\}\cup A'', \{\beta[I](s)\}G''> \\
\bigwedge & & \\
SrCnt[I\}P'' & = & <S'\cup S'' , K[I]\cup A'', \Sigma[I\}G''> \\
\bigwedge & & \\
\{\mu[I](a)\}SrCnt[I\}P'' & = & <S'\cup S'' , \{a\}\cup K[I]\cup A'', \{\mu[I](a)\}\Sigma[I\}G''> \\
\bigwedge & & \\
\underbrace{AlMch[I]\cup SrCnt[I]}_{Dgrm[I]}\cup P'' & = & \underbrace{<S'\cup S'' , A'\cup K[I]\cup A''}_{L[I]}, \underbrace{\Delta[I\}\Sigma[I]}_{\Phi[I]}\cup G''>
\end{array}
$$

Fig. 4.22: Expansiveness by translation diagram: $P'' \preceq Dgrm[I] \cup P''$

7.4 Modularisation construction (many-sorted)

hall now examine the Modularisation Construction, by extending to many-sorted interpertations the ideas seen in 4.3.4 for the unsorted case.

As in figure 4.4, we have (many-sorted) 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 $f:P\rightarrow R$. The Modularisation Construction completes a rectangle of interpretations by amalgamated sum.

As before, 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. The main difference in the many-sorted case lies in the first stage: the Language Modularisation Construction.

The many-sorted Modularisation Construction for languages proceeds as follows (see figure 4.4). We have languages $L_j=<S_j,A_j,V_j>$ with declaration d_j, for j=0,1,2. Without loss of generality, we may assume, by resorting to renaming if necessary, that the underlying languages L_1 and L_2 are disjoint: $L_1\cap L_2=\varnothing$. We shall construct a language $L_3=<S_3,A_3,V_3>$ with declaration d_3 and translation $g:L_1\rightarrow L_3$ in two steps: first sorts (and variables), then alphabet (and declaration).

We first obtain the sets of sorts S_3 and of variables V_3 as before. We set $T:=S_1-S_0$ and $S_3:=S_2\cup T$, then we extend $f_S:S_0\rightarrow S_2$ to $g_S:S_1\rightarrow S_3$ by letting it be the identity on T. Similarly, we put $V_3:=V_2\cup(V_1-V_0)$ and extend $f_V:V_0\rightarrow V_2$ to $g_V:V_1\rightarrow V_3$ by the identity.

We are now ready for the second step: the construction of the alphabet and declaration. We let $B:=A_1-A_0$; for each extra-logical symbol in B we add to A_2 a corresponding symbol b# of the same kind and extend declaration

d_2 by $d_3(b\#):=g_S^*[d_1(b)]$ (see figure 4.23), as well as $f_A:A_0 \rightarrow A_2$ to $g_A:A_1 \rightarrow A_3$ by $g_A(b):=b\#$. We thus have language $L_3:=<S_3,A_3,V_3>$ with declaration d_3. Also, mappings $g_S:S_1 \rightarrow S_3$, $g_V:V_1 \rightarrow V_3$ and $g_A:A_1 \rightarrow A_3$ make the diagrams in figure 4.14 commute, so we have a translation $g:L_1 \rightarrow L_3$ as in figure 4.5.

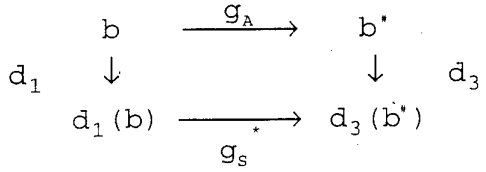

Fig. 4.23: Many-sorted Modularisation: language and translation

This many-sorted Language Modularisation Construction is a special pushout. Notice that, in general, g_S is not injective; nevertheless, g_A is injective on the new symbols in A_1-A_0, even though not the identity (see figure 4.24). The properties of this Language Modularisation Construction are similar to its unsorted version (see 4.3.4), as is the Specification Modularisation Construction: specification $S=<L_3,G_3>$ has as its set of axioms the union $G_3:=G_2 \cup g(G_1)$.

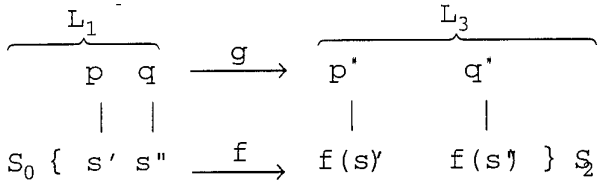

Fig. 4.24: Many-sorted Language Modularisation Construction

4.8 The (many-sorted) Modularisation Theorem

The Modularisation Theorem asserts that the Modularisation Construction preserves conservativeness. As in the case of unsorted specifications, we consider first some variations, including the case of extensions, which as in 4.4.2, guarantees that union preserves conservativeness.

4.8.1 Variations of (many-sorted) modularity

We shall now examine some variations of (many-sorted) modularity: modularity which can be easily seen to carry over from the unsorted case. These concern extensions with special syntactical forms, expansive extensions and extensions by axioms of language L_0 (Axiom Modularity).

We first consider the particular case where the given extension has some special syntactical form. The cases mentioned in 4.4.2.A carry over to the many-sorted case. We now have some further extensions with special syntactical forms, namely the sort introduction constructs seen in 3.8.. But, the arguments in 4.4.2.A extend to cover these new cases as well. The proof of the Modularisation Theorem for such cases is quite

straightforward. Unfortunately, these arguments still rely heavily on the special syntactical form of the new axioms. They do not appear to extend to the general case, when the new axioms are not guarnteed to have any such special form.

Another special case of modularity concerns expansive extensions. In this case, the extensions are not required to have some special syntactical forms, but the previous model-oriented arguments carry over to establish many-sorted preservation of expansiveness:

Orthogonal Expansive Modularity (see 4.4.1.A)

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

Given a family $P_i = \langle L_i, G_i \rangle$ of extensions of $P = \langle L_\cap, G \rangle$, for $i \in I$, let

$P^\cup := \cup_{i \in I} P_i$. For each $j \in I$ such that $P_j \preceq P_k$, for every $k \neq j$ in I, $P_j \preceq P^\cup$.

Expansiveness is a sufficient, but not necessary, condition for conservativeness. So, we have again a special version of the Modularisation Theorem: with stronger hypothesis and conclusion.

Finally, another special version carrying over to the many-sorted case is Axiom Modularity. Notice that its proof in 4.4.1.D relied on the Compactness and Deduction Theorem. We thus have:

Axiom Modularity (see 4.4.1.D)

If $\langle L', G' \rangle \leq \langle L'', G'' \rangle$, then $\langle L', G' \cup H \rangle \leq \langle L'', G'' \cup H \rangle$, for every $H \subseteq Sent(L')$.

.2 Modularity of (many-sorted) extensions

We shall now proceed towards establishing the Modularisation Theorem for (many-sorted) extensions.

Modularity of extensions guarantees that union preserves conservativeness. As such, it resembles a well known logical result related to union of consistent theories: Robinson's Joint Consistency Theorem (Chang and Keisler 1973, p. 88). Recall that a maximally consistent specification is one that is both complete and consistent.

Proposition Robinson's Joint Consistency Theorem

Given sub-languages L_1 and L_2 of L_3, let $L_0 = L_1 \cap L_2$. Consider a specification $P = \langle L_0, G_0 \rangle$, with consistent extensions $Q = \langle L_1, G_1 \rangle$ and $R = \langle L_2, G_2 \rangle$. If specification $P = \langle L_0, G_0 \rangle$ is maximally consistent, then the union specification $Q \cup R = \langle L_1 \cup L_2, G_1 \cup G_2 \rangle$ is consistent.

Proof

The usual proofs, as in (Chang and Keisler 1973), either in the Lindenbaum-Henkin style with witnesses (p. 84-89) or by elementary chains (p. 116-117), can be adapted to the many-sorted case.

QED

Let us compare more closely Robinson's Joint Consistency Theorem and Modularity of Extensions. Robinson's Joint Consistency Theorem guarantees that a property - consistency - of specifications is transmitted to the union specification (over a maximally consistent specification); whereas Modularity of Extensions asserts that a relationship between specifications - conservativeness - is preserved by the union construction. So, the latter appears to be more flexible than the former, in that it replaces consistent extensions of a maximally consistent specification by a conservative extension of a (not necessarily complete) specification. We know that the conservative extensions of a maximally consistent specification are exactly the consistent ones (see 3.4). So, indeed Robinson's Joint Consistency Theorem follows from Modularity of Extensions.

We shall now use Robinson's Joint Consistency Theorem to establish Modularity of (many-sorted) Extensions.

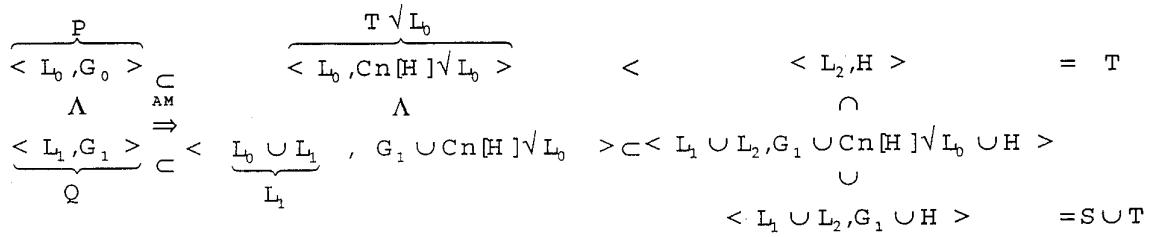

Fig. 4.25: Proof of Modularity of (many-sorted) extensions

Proposition Modularity of (many-sorted) Extensions

Given sub-languages L_1 and L_2 of L_3, let $L_0=L_1\cap L_2$. Consider a specification $P=\langle L_0,G_0\rangle$, with extensions $Q=\langle L_1,G_1\rangle$ and $R=\langle L_2,G_2\rangle$. If Q is a conservative extension of P ($P\leq Q$), then the union specification $S=\langle L_1\cup L_2,G_1\cup G_2\rangle$ is a conservative extension of R: $R\leq Q\cup R$.

Proof (see figure 4.25)

Let $S:=Q\cup R$. Given a maximally consistent specification $T=\langle L_2,H\rangle$ extending $R=\langle L_2,G_2\rangle$, we will show that $S\cup T=\langle L_1\cup L_2,G_1\cup G_2\cup H\rangle$ is consistent.

For this purpose, consider the restriction $T\sqrt{L_0}:=\langle L_0,Cn[H]\sqrt{L_0}\rangle$ of $T=\langle L_2,H\rangle$ to sub-language $L_0\subseteq L_3$, where $Cn[H]\sqrt{L_0}:=\{\sigma\in Sent(L_0)/H\models\sigma\}$.

We then have a conservative extension $T\sqrt{L_0}\leq T$ (see 3.3), with $P\subseteq T\sqrt{L_0}$ since $P\subseteq R\subseteq T$. So, $P\subseteq T\sqrt{L_0}\leq T$.

Thus $\langle L_0,Cn[H]\sqrt{L_0}\rangle\equiv\langle L_0,G_0\cup Cn[H]\sqrt{L_0}\rangle$, and Axiom Modularity yields $\langle L_0,Cn[H]\sqrt{L_0}\rangle\leq\langle L_1,G_1\cup Cn[H]\sqrt{L_0}\rangle$.

Since T is maximally consistent, so is $T\sqrt{L_0}$; entailing the consistency of its conservative extension $\langle L_1,G_1\cup Cn[H]\sqrt{L_0}\rangle$.

Hence, Robinson's Joint Consistency Theorem yields the consistency of $<L_1 \cup L_2, G_1 \cup Cn[H] \sqrt{L_0 \cup H}>$, and hence that of $S \cup T = <L_1 \cup L_2, G_1 \cup G_2 \cup H>$.

QED

3 Modularity of (many-sorted) interpretations

We shall now proceed towards a proof of the (many-sorted) Modularisation Theorem. We could adapt the proof for its unsorted version, but we prefer to present a sligthly different proof, which illustrates the application of the technique of diagram internalisation. We shall reduce it to Modularity of Extensions by means of (many-sorted) diagram internalisation.

The diagram of $I:L' \rightarrow L''$ provides the information of the translation in the language $L[I]$ extending the union language $L' \cup L''$. As such, it can be used to extend specifications over a sub-language L of the alphabet matching language $L[I]$. We will show that translation diagrams code the information for completing the pushout rectangle of language translations in the Modularisation Construction. The idea is that each new symbol $l^{\#}$ added in extending L_2 to L_3 is explicable in terms of its originator l by means of the corresponding diagram sentence.

Lemma Diagrams of modular translations

Given a translation $f:L_0 \rightarrow L_2$ and an extension $e:L_0 \subseteq L_1$, let $g:L_1 \rightarrow L_3$ be the canonical extension of f provided by the Modularisation Construction. We then have an expansive extension $<L_1 \cup K[f] \cup L_2, \Phi[f]> \preceq <L_1 \cup K[g] \cup L_3, \Phi[g]> = Dgrm[g]$. In particular, $Dgrm[f] \preceq Dgrm[g]$.

Proof (see figure 4.26)

Letting $T = <L_1 \cup K[f] \cup L_2, \Phi[f]>$, we will show $T \preceq T \cup SrCnt[g] \preceq Dgrm[g]$.

A symbol $l^{\#}$ of $L_1 \cup L_3$ not in $L_1 \cup L_2$ is a symbol of L_3 not in L_2, with, by the Language Modularisation Construction (see 4.3.4), a unique symbol l of $L_1 - L_0$ such that $g(l) = l^{\#}$.

We first show $T \preceq T \cup SrCnt[g]$

For each sort $s^{\#} \in (S_3 - S_2)$, we have $s \in (S_1 - S_0)$ with $g(s) = s^{\#}$ and an expansive extension from T to $T \cup \{\beta[g](s)\}$, the latter over language $L_s := <S_1 \cup S_2 \cup \{s^{\#}\}, A_1 \cup J[f] \cup A_2>$. So, for all $s^{\#} \neq t^{\#}$ in $(S_3 - S_2)$, $L_s \cap L_t = L_1 \cup K[f] \cup L_2$. Thus, orthogonal expansiveness (see 4.7.1.B) yields, as desired, that $T \preceq <L_1 \cup K[g] \cup L_2, \Phi[f] \cup \Sigma[g]> = T \cup SrCnt[g]$.

We now establish $T \cup SrCnt[g] \preceq Dgrm[g]$.

Each predicate or operation symbol $a^{\#} \in (A_3 - A_2)$ has $a \in (A_1 - A_0)$ with $g(a) = a^{\#}$, and $T \cup SrCnt[g] \cup \{\mu[g](a)\}$ is equivalent to an extension by definition of

$T \cup SrCnt[g] = \langle L_1 \cup K[g] \cup L_2, \Phi[f] \cup \Sigma[g] \rangle$ to $L_S := \langle S_1 \cup S_3, A_1 \cup J[g] \cup A_2 \cup \{a^\#\} \rangle$.
For all $a^\# \neq b^\#$ in $(A_3 - A_2)$, $L_a \cap L_b = L_1 \cup K[g] \cup L_2$. Thus, orthogonal expansiveness (see 4.7.1.B) yields $T \cup SrCnt[g] \preceq \langle L_1 \cup K[g] \cup L_3, G \cup \Phi[g] \cup \Sigma[g] \rangle = Dgrm[g]$.
{ Alternatively, we use $\Sigma[g]$ to expand $\mathfrak{A} \in Mod[T]$ to $\mathfrak{B} \in Mod[T \cup SrCnt[g]$, and then $\Delta[g]$ to expand $\mathfrak{B}$ to $\mathfrak{C} \in Mod[Dgrm[g]]$. }
QED

$$L_1 \cup K[f] \cup L_2 \;:\; L_1 \left\{ \begin{matrix} A_1 \\ S_1 \supset \underbrace{S_0 \xrightarrow{J[f]} S_2}_{K[f]} \end{matrix} \right. A_2 \Big\} \; L_2$$

$\Sigma[g] \;\cap$

$$L_1 \cup K[g] \cup L_2 \;:\; L_1 \left\{ \begin{matrix} A_1 \\ \underbrace{S_1 \xrightarrow{J[g]} S_3}_{K[g]} \end{matrix} \right. \supset \begin{matrix} A_2 \\ S_2 \end{matrix} \Big\} \; L_2$$

$\Delta[g] \;\cap$

$$L_1 \cup K[g] \cup L_3 \;:\; L_1 \left\{ \begin{matrix} A_1 \\ \underbrace{S_1 \xrightarrow{J[g]} S_3}_{K[g]} \end{matrix} \right. A_3 \Big\} \; L_3$$

Fig. 4.26: Languages in diagrams of modular translations

Theorem Modularity of (many-sorted) 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 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 4.27)
We will construct a conservative extension $R \leq T$, so that T extends R.
1. Since $f: P \rightarrow R$, by item (b) of the proposition characterising interpretations by translation diagrams, we have $R \leq Dgrm[f] \cup R \supseteq P$.
2. Since $L[f] = L_0 \cup K[f] \cup L_2$, we have $L_1 \cap L[f] = L_0$ and $L_1 \cup L[f] = L_1 \cup K[f] \cup L_2$.
So Modularity of (many-sorted) Extensions yields
$Dgrm[f] \cup R \leq \langle L_1 \cup K[f] \cup L_2, \Phi[f] \cup G_2 \cup G_1 \rangle = Dgrm[f] \cup R \cup Q$.
3. By the preceding lemma, $\langle L_1 \cup K[f] \cup L_2, \Phi[f] \rangle \leq \langle L_1 \cup K[g] \cup L_3, \Phi[g] \rangle = Dgrm[g]$.
So, Axiom Modularity yields the conservativeness

$<L_1 \cup K[f] \cup L_2, \Phi[f] \cup G_2 \cup G_1> \leq <L_1 \cup K[g] \cup L_3, \Phi[g] \cup G_2 \cup G_1>$, i. e.
$Dgrm[f] \cup R \cup Q \leq Dgrm[g] \cup R \cup Q$.

4. Now, by item (b) of the proposition on diagram and translation $g(Q) \subseteq Dgrm[g] \cup Q$. Thus, $S = g(Q) \cup R \subseteq Dgrm[g] \cup Q \cup R$.

5. Therefore $R \leq Dgrm[f] \cup R \leq Dgrm[f] \cup R \cup Q \leq Dgrm[g] \cup Q \cup R$, with $R \leq Dgrm[g] \cup Q \cup R$.

QED

$$\begin{array}{ccccc} \overbrace{<L_0, G_0>}^{P} & \underset{EM}{\subset} & < \overbrace{L_0 \cup K[f] \cup L_2}^{L[f]} , \Phi[f] \cup G_2 > & < & \overbrace{<L_2, G_2>}^{R} \\ \Lambda & \Rightarrow & \Lambda & & \\ \underbrace{<L_1, G_1>}_{Q} & \subset & <L_1 \cup K[f] \cup L_2, \Phi[f] \cup G_2 \cup G_1> & = & Dgrm\ [f] \cup R \cup Q \end{array}$$

$$\begin{array}{ccc} <L_1 \cup K[f] \cup L_2, \Phi[f]> & & \overbrace{<L_1 \cup K[f] \cup L_2, \Phi[f] \cup G_1 \cup G_2>}^{Dgrm\ [f] \cup R \cup Q} \\ \Lambda & \underset{AM}{\Rightarrow} & \Lambda \\ <L_1 \cup K[g] \cup L_3, \Phi[g]> & & \underbrace{<L_1 \cup K[g] \cup L_3, \Phi[g] \cup G_1 \cup G_2>}_{Dgrm\ [g] \cup R \cup Q} \end{array}$$

$$\begin{array}{ccc} Dgrm\ [g] \cup Q & & Dgrm\ [g] \cup Q \cup R \\ \cup & \Rightarrow & \cup \\ g(Q) & & \underbrace{g(Q) \cup R}_{R} \end{array}$$

Fig. 4.27: Proof structure for (many-sorted) Interpretation Modularity

Variations of interpretation

A variant of translation/interpretation sometimes encountered (Turski and Maibaum 1987, p. 265; Maibaum *et al.* 1991, p. 14) involves mapping a sort to a sequence of sorts with relativisation predicates and translation of equality. This is motivated by some needs in implementatation. We shall examine such general variants and see that they can be reduced to simple interpretations with introduction of product sorts, subsorts and quotient sorts.

At the beginning of 3.8 we mentioned the overused implementation of stacks by arrays with indices. Here one wishes to represent a stack by means of an array, containing the elements stored in the stack, together with an index, indicating the position of the topmost element. Thus, each stack s is to be represented by a pair <a,i>; but, not every such pair will

represent a stack; furthermore distinct such pairs may represent the same stack. This suggests translating sort Stk to sequence <Arr,Ind> of sorts, with a (binary) relativisation predicate, over sorts Arr,Ind, as well as a relation, over sorts Arr,Ind,Arr,Ind, to represent equality of stacks.

As mentioned in 4.7.2, the decompositions induced by language partitions seen in 4.3.2 can be extended to the many-sorted case. This will enable us to decompose such general many-sorted interpretations into somewhat simpler variants, thereby decoupling the effect of each generalisation. We shall successively consider relativisation (which we will reduce to subsort introduction), translation of equality (reducible to quotient sort) and sort tupling (replaceable by introduction of a product sort). Then, we shall examine such general interpretations incorporating the three features and indicate how they can be reduced to simple interpretations into extensions by introduction of appropriate sorts.

4.9.1 Relativisation and subsort

We have briefly examined unsorted translations/interpretations involving relativisation predicates in 4.5.4. Let us now consider their many-sorted versions (Turski and Maibaum 1987, p. 265; Maibaum *et al.* 1991, p. 14). Our introductory example in 4.1 motivates their need: not every target object represents a source object. We shall concentrate on the effects of relativisation and then indicate how such interpretations can be reduced to simple interpretations into subsorts.

Consider many-sorted languages L' and L"; a *translation* $I:L' \rightarrow L''$ *with relativisation predicates* assigns to each sort $s \in Srt(L')$ both a sort $I_S(s)=t_s \in Srt(L'')$ (the *translation of sort* s) and a unary predicate symbol $I_r(s)=r_s \in Prd(L'')$, over sort t_s (called the *relativisation predicate for sort* s). The idea is, as before, that r_s represents within L" the universe of sort s of L'. The effect of this in translating is felt only in the quantified formulae of L'. A formula $\varphi \in Frml(L')$ is mapped to $\varphi^r \in Frml(L'')$ by relativising quantifiers: with $w=I_V(v)$, $[(\forall v:s)\theta]^r:=(\forall w:t_s)[r_s(w) \rightarrow \theta^r]$ and $[(\exists v:s)\theta]^r:=(\exists w:t_s)[r_s(w) \wedge \theta^r]$. Finally, we obtain the translation of a formula by relativising its free variables: $I(\varphi):=[r_{s_1}(w_1) \wedge \ldots \wedge r_{s_m}(w_m) \rightarrow \varphi^r]$, where $v_1:s_1,\ldots,v_m:s_m$ are the free variables of φ, and $w_1,\ldots,w_m$ their respective translations. Notice that our simple translations may be regarded as special cases of translations with relativisation predicates: those with trivial relativisation predicates $w \approx w$. In fact, relativisation for only some sorts is the special case of translation with relativisation predicates where the remaining sorts have trivial relativisation predicates.

For the reasons mentioned in 4.5.4.A, some care is necessary for such translations to exhibit proper behaviour. Let us assume for the moment

that L' has no operation symbols: $Opr(L')=\emptyset$. We then restrict such translations with relativisation predicates as follows. Given a specification $P''=<L'',G''>$, we say that translation $I:L'\rightarrow L''$, assigning to each sort s of L' a sort t_s and relativisation predicate r_s of L", is an *interpretation* $I:<L',\emptyset>\rightarrow<L'',G''>$ *with relativisation predicates* iff the non-voidness requirements $(\exists w:t_s)r_s(w)$ are in $Cn[P'']$ for all $s\in Srt(L')$ (in case $Opr(L')=\emptyset$).

With these requirements we achieve induction of structures: each model $M\in Mod[P'']$ induces a structure $\mathfrak{M}\downarrow I$ for language L', with universe of sort $s\in Srt(L')$ being the realisation of the relativisation predicate in the given structure $\mathfrak{M}$: $\mathfrak{M}\downarrow I[s]=\mathfrak{M}[r_s]$ (notice that $\mathfrak{M}[r_s]\neq\emptyset$).

Now, an assignment a" of values in $\mathfrak{M}$ to the variables of L" may assign to a variable v over sort t_s in L" an object in $\mathfrak{M}[s]$ outside $\mathfrak{M}[r_s]$. To single out assignments for L" that can be induced by assignments for L' we consider the idea of conjugate assignments: assignments a' in $\mathfrak{M}\downarrow I$ to the variables L' and a" in $\mathfrak{M}$ to the variables of L" are called *conjugate* (denoted $a'\sim a''$) iff for each variable $v\in Var(L')$ $a'(v)=a''[I_V(v)]$. We then have a Translation Connection: $\mathfrak{M}\downarrow I\models\varphi[a']$ iff $\mathfrak{M}\models I(\varphi)[a'']$, for each formula $\varphi\in Frml(L')$ and conjugate assignments $a'\sim a''$.

Recall from 3.8.3 that the non-voidness requirement $\nu(r)$ is what is required for introducing a subsort. Thus, subsort introduction provides both an illustrative example of interpretations with relativisation predicates and a way of reducing them to simple interpretations with the same effect.

First, consider a language L with sorts s and t, a unary predicate symbol r over sort t. Now, consider the sub-languages of L: $L':=\{s\}$ and $L'':=\{t,r\}$. By assigning to sort s of L' both sort t and predicate r of L", we obtain an interpretation with relativisation predicate $J:<L',\emptyset>\rightarrow<L'',\{(\exists w:t)r(w)\}>$.

Consider now a specification $P''=<L'',G''>$ such that $P''\models(\exists w:t)r(w)$. We can then introduce a new sort d as the subsort of t under relativisation predicate r, obtaining a specification $P^{\subseteq}=SR\_SBST[d\backslash t:r,j](P'')$, with language $L^*=L''\cup\{d,j(d)\rightarrow t\}$ and axiomatisation $G^{\subseteq}=G''\cup\{\subseteq(d\backslash t:r,j)\}$, which is an expansive extension of P". Now, given an interpretation $I:<L',\emptyset>\rightarrow<L'',G''>$, assigning to sort s of L' both sort t and relativisation predicate r of L", we can obtain a simple interpretation $I^{\subseteq}:<L',\emptyset>\rightarrow P^{\subseteq}$ by mapping sort s directly to the new subsort d. Conversely, given such a simple interpretation $I^{\subseteq}:<L',\emptyset>\rightarrow P^{\subseteq}$, mapping sort s of L' to the subsort d of t relativised to predicate r in L", we obtain from it an interpretation $I:<L',\emptyset>\rightarrow P''$ with relativisation predicate, by mapping sort s of L' to sort t and relativisation

predicate r in L". Moreover, in view of the properties of subsort under connection in 3.8.3, specification P"=<L",G"> and its expansive extension $P^{\subseteq}$ = <L"$\cup${d,j(d)$\rightarrow$t},G"$\cup${$\subseteq$(d\t:r,j)}> have the same expressive and deductive powers, being thus interchangeable.

Now, let us examine the case where source language L' has operation symbols. We now place more stringent restrictions on such translations with relativisation predicates, as follows. Given specifications P'=<L',G'> and P"=<L",G">, we say that translation I:L'$\rightarrow$L", assigning to each sort s of L' a sort t_s and relativisation predicate r_s of L", is an *interpretation* I:P'$\rightarrow$P" *with relativisation predicate* iff, in additon to the translation I(G') of the axioms of P', the following requirements are in Cn[P"]:

- the non-voidness requirements $\nu(r_s)$: $(\exists w{:}t_s)r_s(w)$ for each source sort s;
- for each operation f$\in$ Opr(L'), from sorts $s_1,\dots,s_n$ to s, the closure requirement $\chi(r_{s_1},\dots,r_{s_n}\rightarrow r_s)[g]$ of $r_{s_1},\dots,r_{s_n}$ and r_s with respect to g=I(f):

$$(\forall w_1{:}t_1)\dots(\forall w_n{:}t_n)[(r_{s_1}(w_1)\wedge\dots\wedge r_{s_n}(w_n))\rightarrow r_s(g(w_1,\dots,w_n))].$$

Notice that for a model $\mathfrak{M}\in$ Mod[P"] and an operation symbol f$\in$ Opr(L'), from sorts $s_1,\dots,s_n$ to s in L', $\mathfrak{M}\models\chi(r_{s_1},\dots,r_{s_n}\rightarrow r_s)[I(f)]$. So, $\mathfrak{M}$[I(f)] maps $\mathfrak{M}[r_{s_1}]\times\dots\times\mathfrak{M}[r_{s_n}]$ into $\mathfrak{M}[r_s]$, thus inducing a function $\mathfrak{M}{\downarrow}I[f]{:}\mathfrak{M}{\downarrow}I[s_1]\times\dots\times\mathfrak{M}{\downarrow}I[s_1]\rightarrow\mathfrak{M}{\downarrow}I[s]$.

As above, if P"$\models\nu(r_s)$ for predicates r_s over sorts t_s, we can extend P"=<L",G"> to $P^{\subseteq}$, by introducing subsorts d_s together with conversions $j_s(d_s)\rightarrow t_s$, axiomatised by $\subseteq(d_s\backslash t_s{:}r_s,j_s)$. Also, each operation g$\in$ Opr(L"), from sorts $t_{s_1},\dots,t_{s_n}$ to t_s, such that P"$\models\chi(r_{s_1},\dots,r_{s_n}\rightarrow r_s)[g]$ gives rise to an operation g^r, involvng the corresponding subsorts $d_{s_1},\dots,d_{s_n}$ and d_s, introduced by the defining axiom

$$(\forall x_1{:}d_{s_1})\dots(\forall x_n{:}d_{s_n})(\forall y{:}d_s)[g^r(x_1,\dots,x_n)\approx y\leftrightarrow g(j_{s_1}(x_1),\dots,j_{s_n}(x_k)))\approx j_s(y)].$$

Similarly, predicates involving sorts $t_1,\dots,t_m$ give rise to predicates over the subsorts $d_{s_1},\dots,d_{s_m}$. We thus have an extension P^r by definitions of $P^{\subseteq}$, where we can interpret P' via a simple translation I^r, with the same effect. Hence, such many-sorted interpretations with relativisation predicates - and operation and predicate symbols - can be reduced to simple interpretations into subsorts.

4.9.2 Translation of equality and quotient sorts

We shall now consider another variant of many-sorted translations/interpretations, those involving translation of equality (Turski and Maibaum 1987, p. 163), whose unsorted version was briefly examined in 4.5.4. Our example in 4.1 gives the motivation for this: several target

objects may represent the same source object. We shall concentrate on the effects of translation of equality and then indicate how such interpretations can be reduced to simple interpretations into quotient sorts.

Consider many-sorted languages L' and L"; a *translation* $I:L' \rightarrow L''$ *of equality* assigns to sort s both a sort $I_S(s)=t_s \in Srt(L'')$ (the *translation of sort* s) and a binary predicate symbol $I_q(s)=q_s \in Prd(L'')$, over sorts t,t (called the *translation of equality over sort* s). The idea is that q_s represents within L" the identity over the universe of sort s of L'. This affects only the translation of equality formulae: an atomic formula $t \approx t'$, with t and t' terms of sort s in L', is translated to $q_s(t,t')$ (see, e. g. Turski and Maibaum 1987, p. 163). Notice that our simple translations correspond to special cases of translations of equality - those with trivial translations of equality $u \approx v$. Also, we can translate equalities only between terms of some sorts by assigning to the remaining ones trivial translations of equality.

Some precautions are necessary, due to reasons similar to those indicated in 4.5.4.B, for proper behaviour of such translations. In order to examine them, let us assume for the moment that L' has no operation or predicate symbols, other than equality: $Opr(L')=\varnothing$ and $Prd(L')=\varnothing$. We are then led to considering restrictions on such translations of equality as follows. Given a specification $P''=<L'',G''>$, we say that translation of equality $I:L' \rightarrow L$, assigning to sort s of L' sort t_s and binary predicate q_s of L", is an *interpretation* $I:<L',\varnothing> \rightarrow <L'',G''>$ *with translation of equality* iff the equivalence requirement $\varepsilon(q_s)$ are in Cn[P"] (in case $Opr(L')=\varnothing=Prd(L')$).

With these requirements we achieve the desired proper behaviour. In particular, each model $\mathfrak{M} \in Mod[P'']$ induces a structure $\mathfrak{M}\downarrow I$ for language L', with $\mathfrak{M}\downarrow I[s]=\mathfrak{M}[t_s]/\mathfrak{M}[q_s]$. This gives rise to a Translation Connection as follows: $\mathfrak{M}\downarrow I \models \varphi\ [a']$ iff $\mathfrak{M} \models I(\varphi)\ [a]$, for each formula $\varphi \in Frml(L')$ and assignment a of values in $\mathfrak{M}$ to the variables of L", inducing assignment a' of values in $\mathfrak{M}\downarrow I$ to the variables of L' by $a'(v)$ being the q_s-class of a(v).

But, we know from 3.8.4 that in the presence of the equivalence requirement we can introduce a new sort d as the quotient of sort t under equivalence predicate q. Much as in the case of relativisation predicates, such interpretations of equality are reducible to simple interpretations into an extension by introduction of a quotient sort, because of the properties of quotient sort under connection in 3.8.4.

Now, let us turn to the general case where source language L' has predicate and operation symbols. Then, it is wise to place more stringent restrictions on such translations of equality, as follows. Given specifications

P'=<L',G'> and P"=<L",G">, we say that translation of equality I:L'→L", assigning to each sort s of L' a sort t_s and a binary predicate q_s of L", is an *interpretation* I:P'→P" *with translations of equality* iff, in additon to the translation I(G') of the axioms of P', the following requirements are in Cn[P"]:

- the equivalence requirements $\varepsilon(q_s)$ (see 3.8.4); as well as
- the congruence requirements of q_s with respect to
  - the translation g=I(f) of each operation f∈ Opr(L'), say from sorts $s_1,\dots,s_n$ to s in L', i.e. the sentence $\kappa(q_{s_1},\dots,q_{s_n}\rightarrow q_s)[g]$:

$$(\forall u_1,v_1{:}t_{s_1})\dots(\forall u_n,v_n{:}t_{s_n})[(q_{s_1}(u_1,v_1)\wedge\dots\wedge q_{s_n}(u_n,v_n))\rightarrow \rightarrow q_s(g(u_1,\dots,u_n),g(v_1,\dots,v_n))];$$

  - the translation r=I(p) of each predicate p∈ Prd(L'), say over sorts $s_1,\dots,s_m$ in L', i.e. the sentence $\kappa(q_{s_1},\dots,q_{s_m})[r]$:

$$(\forall u_1,v_1{:}t_{s_1})\dots(\forall u_m,v_m{:}t_{s_m})[(q_{s_1}(u_1,v_1)\wedge\dots\wedge q_{s_m}(u_m,v_m))\rightarrow \rightarrow(r(u_1,\dots,u_m)\rightarrow r(v_1,\dots,v_m))].$$

Notice that for a model $\mathfrak{M}\in$ Mod[P"] and an operation symbol f∈ Opr(L'), from sorts $s_1,\dots,s_n$ to s in L', translated to g=I(f), $\mathfrak{M}\models\kappa(q_{s_1},\dots,q_{s_n}\rightarrow q_s)[g]$. So, whenever $<a_1,b_1>\in\mathfrak{M}[q_{s_1}],\dots,<a_n,b_n>\in\mathfrak{M}[q_{s_n}]$, we also have $<g(a_1,\dots,a_n),g(b_1,\dots,b_n)>\in\mathfrak{M}[q_s]$, which enables the induction of a function $\mathfrak{M}\downarrow I[f]:\mathfrak{M}\downarrow I[s_1]\times\dots\times\mathfrak{M}\downarrow I[s_1]\rightarrow\mathfrak{M}\downarrow I[s]$. Similarly for predicate symbols, $\mathfrak{M}[I(p)]$ induces $\mathfrak{M}\downarrow I[p]$ defined on equivalence classes.

When P"$\models\varepsilon(q_s)$ for predicates q_s over sorts t_s,t_s, we can extend P"=<L",G"> to P/, by introducing quotient sorts d_s together with projections $p_s(t_s)\rightarrow d_s$, axiomatised by $/(d_s\backslash t_s/q_s,p_s)$, which is an expansive extension of P". Also, each predicate r∈ Prd(L"), over sorts $t_{s_1},\dots,t_{s_m}$, such that P"$\models\kappa(q_{s_1},\dots,q_{s_m})[r]$ gives rise to a predicate r^q, over quotient sorts $d_{s_1},\dots,d_{s_m}$, introduced by the defining axiom

$$(\forall w_1{:}d_{s_1})\dots(\forall w_m{:}d_{s_m})\{r^q(w_1,\dots,w_m)\leftrightarrow \leftrightarrow(\exists u_1{:}t_{s_1})\dots(\exists u_m{:}t_{s_m})[p_{s_1}(u_1)\approx w_1\wedge\dots\wedge p_{s_m}(u_m)\approx w_1\wedge r(w_1,\dots,w_m)]\}.$$

Furthermore, each operation g∈ Opr(L"), from sorts $t_{s_1},\dots,t_{s_n}$ to t_s, such that P"$\models\kappa(q_{s_1},\dots,q_{s_n}\rightarrow q_s)[g]$ induces an operation g^q, involving the corresponding quotient sorts $d_{s_1},\dots,d_{s_n}$ and d_s, introduced by the axiom

$$(\forall x_1{:}d_{s_1})\dots(\forall x_n{:}d_{s_n})(\forall y{:}d_s)\{g^q(x_1,\dots,x_n)\approx y\leftrightarrow \leftrightarrow(\exists z_1{:}t_{s_1})\dots(\exists z_n{:}t_{s_n})[p_{s_1}(z_1)\approx x_1\wedge\dots\wedge p_{s_n}(z_n)\approx x_n\wedge y\approx p(g(x_1,\dots,x_n))].$$

We thus have an extension P^q by definitions of P/, where we can interpret P' via a simple translation I^q with the same effect. Hence, such many-sorted interpretations with translation of equality - and operation and

predicate symbols - can be reduced to simple interpretations into quotient sorts.

4.9.3 Sort tupling and product sort

Now, let us examine many-sorted translations/interpretations involving sort tupling (Turski and Maibaum 1987, p. 265; Maibaum *et al.* 1991, p. 14). Our example of implementing stacks by arrays with indices serves as motivation: a source sort is represented by a sequence of target sorts. We shall concentrate on the effects of sort tupling and indicate how such interpretations can be reduced to simple interpretations into product sorts.

Let us examine such translations mapping a sort to a sequence of sorts. For the sake of simplicity we start with the case where the source language L' has no operation or predicate symbols, other than equality: $Opr(L')=\varnothing$ and $Prd(L')=\varnothing$.

Consider many-sorted languages L' and L"; a *translation* $I:L'\rightarrow L"$ *with sort tupling* assigns to each sort $s\in Srt(L')$ a non null sequence $I_t(s)=\langle s^1,\ldots,s^k\rangle$ of sorts of L", called the *sort tupling* of s. The idea is that the k-tuple $\langle s^1,\ldots,s^k\rangle$ represents within L" the universe of sort s of L'. Since we wish to translate terms and formulae, sort tupling comes accompanied by a *variable tupling* assigning to each variable $v\in Var(L')$, ranging over sort s of L', a distinct k-tuple $I_V(v)=\langle v^1,\ldots,v^k\rangle$ of variables of L", with $v^1,\ldots,v^k$ ranging over $s^1,\ldots,s^k$ respectively. We shall generally denote both maps I_x and I_V simply by I. Notice that our simple translations may be viewed as special cases of translations with sort tupling: those with trivial sort tupling $I_t(s)=\langle t\rangle$.

With these data we can translate terms and formulae of L' to L" by following the same idea as before: replacement of symbols of L' by their corresponding ones in L". But a variable v of L' gets translated to a sequence $\langle v^1,\ldots,v^k\rangle$ of variables of L"; and this affects the translation of two kinds of formulae of L':

equalities between terms of sort s - an atomic formula $v\approx w$ of L' is translated to the conjunction $v^1\approx w^1\wedge\ldots\wedge v^k\approx w^k$;

formulae with quantification over sort s - $(Qv:s)\theta$ is translated to $(Qv^1:s^1)\ldots(Qv^k:s^k)I(\theta)$;

where $\langle v^1,\ldots,v^k\rangle=I_V(v)$ and $\langle w^1,\ldots,w^k\rangle=I_V(w)$.

A structure $\mathfrak{M}$ for L" induces a $\mathfrak{M}\downarrow I$ structure for L' keeping the idea that $\langle s^1,\ldots,s^k\rangle$ represents within L" the universe of sort s of L': $\mathfrak{M}\downarrow I[s]$ is the Cartesian product $\mathfrak{M}[s^1]\times\ldots\times\mathfrak{M}[s^k]$. Similarly, an assignment a" to variables of sorts $s^1,\ldots,s^k$ of L" into $\mathfrak{M}$ and an assignment a' to variables of sort s of L' into $\mathfrak{M}\downarrow I$ correspond to each other under $a'(v)=\langle a"(v^1),\ldots,a"(v^k)\rangle$. So, the

Translation Connection is formulated as: $\mathfrak{M}\downarrow I \models \varphi\ [a']$ iff $\mathfrak{M} \models I(\varphi)\ [a'']$ for each formula $\varphi \in Frml(L')$ and assignments such that a' and a" correspond to each other.

The above considerations are based on the idea of representing a sort as a Cartesian product of sorts. This is exactly the idea of introducing a product sort, as in 3.8.1. So, introduction of a product sort provides both an illustrative example of translations with sort tupling and a way of reducing them to simple interpretations with the same effect. We proceed to clarify this connection, beginning with the case without operation or predicate symbols, other than equality.

First, consider a language L with sorts $s^1,\ldots,s^k$ and s. Now, consider the sub-languages of L: L':={s} and L":={$s^1,\ldots,s^k$}. By assigning to sort s of L' the k-tuple $<s^1,\ldots,s^k>$ of sorts of L", we obtain a translation with sort tupling by providing an appropriate tupling of variables.

Consider a language L" with sorts $s^1,\ldots,s^k$, and introduce a new sort t as the n-fold product of these sorts with projections $p^i:t\rightarrow s^i$. This provides an extended specification $P^x=SR\_PROD[t\backslash p^1:s^1,\ldots,p^k:s^k](<L,\varnothing>)$, with language $L^x=L\cup\{t,p^1:t\rightarrow s^1,\ldots,p^k:t\rightarrow s^k\}$. Recall that specification $P^x=<L^x,G^x>$ is an expansive extension of $<L",\varnothing>$ with eliminability under connection (see 3.8.1). Now, consider a translation $I:L'\rightarrow L"$ with sort tupling, assigning to sort $s\in Srt(L')$ a k-tuple $I_t(s)=<s^1,\ldots,s^k>$ of sorts of L", accompanied by a variable tupling I_V. We can redefine I to map sort s of L' to the product sort t, redefining appropriately the renaming of variables. This gives a simple translation $I^x:L'\rightarrow L^x$ interpreting $<L',\varnothing>$ into $<L^x,G^x>$. Conversely, given such a simple interpretation $I^x:<L',\varnothing>\rightarrow<L^x,G^x>$, we obtain from it a translation $I:L'\rightarrow L"$ with sort tupling, assigning to sort $s\in Srt(L')$ the k-tuple $I_t(s)=<s^1,\ldots,s^k>$ of sorts of L", accompanied by a variable tupling I_V. Moreover, in view of the eliminability of a product sort under connection 3.8.1, $<L",\varnothing>$ and its expansive extension $<L^x,G^x>$ have the same expressive and deductive powers, being thus interchangeable.

Let us now turn to the general case where source language L' has predicate and operation symbols. Since a sort s of L' is mapped to a sequence of sorts of L", we must translate predicates and operations involving sort s accordingly. So, a *translation* $I:L'\rightarrow L"$ *with sort tupling* supplements the sort tupling I_t and injective variable tupling I_V with the following renamings:

predicate renaming $I_R:R'\rightarrow R"$, assigning to each predicate symbol $r\in Prd(L')$, over sorts $s_1,\ldots,s_m$ in L', a predicate symbol $I_R(r)\in Prd(L")$,

over sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_m^1,\ldots,s_m^{k_m}$ in L", where

$<s_1^1,\ldots,s_1^{k_1}>=I_t(s_1),\ldots,<s_m^1,\ldots,s_m^{k_m}>=I_t(s_m)$;

operation renaming $I_F:F'\rightarrow F"$, assigning to each operation symbol $f\in Opr(L')$, from sorts $s_1,\ldots,s_n$ to s in L', with $I_t(s)=<s^1,\ldots,s^k>$, k operation symbols $f^1,\ldots,f^k\in F"=Opr(L")$, each f^i being an operation from sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_n^1,\ldots,s_n^{k_n}$ to sort s^i, of L", where

$<s_1^1,\ldots,s_1^{k_1}>=I_t(s_1),\ldots,<s_n^1,\ldots,s_n^{k_n}>=I_t(s_n)$.

As before, we have translations: of terms and formulae:

- with a term $t\in Trm(L')[s]$ of L' being translated to a k-tuple $I(t)=<t^1,\ldots,t^k>\in Trm(L")[s^1]\times\ldots\times Trm(L")[s^k]$ of terms of L"; and
- a formula $\varphi\in Frml(L')$ being translated to a formula $I(\varphi)\in Frml(L")$.

We call such a translation $I:L'\rightarrow L"$ with sort tupling an *interpretation with sort tupling* from P'=<L',G'> into P"=<L",G"> iff the translation of each theorem of P' is a theorem of P": $I(Cn[P'])\subseteq Cn[P"]$.

We can now extend the previous argument to include predicates and operations as follows. Consider language L" with sorts $s_i^1,\ldots,s_i^{k_i}$ and introduce new sorts t_i as the k_i-fold product of these sorts with projections $p_i^{k_i}:t_i\rightarrow s_i^{k_i}$. This gives an expansive extension $P^\times$ of P", with eliminability under connection. Now, each predicate $r\in Prd(L")$, over sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_m^1,\ldots,s_m^{k_m}$ of L", gives rise to a predicate r^t, over sorts $t_1,\ldots,t_m$ of L, introduced by the defining axiom:

$$(\forall v_1:t_1)\ldots(\forall v_m:t_m)[r^t(v_1,\ldots,v_m)\leftrightarrow r(p_1^1(v_1),\ldots,p_m^{k_m}(v_m))].$$

Also, each each k-tuple $g=<g^1,\ldots,g^k>$ of operation symbols $g^i\in Opr(L")$, each g^i being an operation from sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_n^1,\ldots,s_n^{k_n}$ to sort s^i, induces an operation g^t, from sorts $t_1,\ldots,t_n$ to t, introduced by the axiom

$$(\forall x_1:t_1)\ldots\ldots(\forall x_n:t_n)(\forall y:t)\{g^t(x_1,\ldots,x_n)\approx y\leftrightarrow$$
$$\leftrightarrow[p^1(y)\approx g^1(p_1^1(x_1),\ldots,p_n^{k_n}(x_n))\wedge\ldots\wedge p^k(y)\approx g^k(p_1^{k_n}(x_n),\ldots,p_n^{k_n}(x_n))]\}.$$

We thus have an extension P^t by definitions of $P^\times$, where we can interpret $<L',\varnothing>$ via a simple translation I^t with the same effect. Hence, such general many-sorted interpretations with sort tupling, as well as operation and predicate symbols, can be reduced to simple interpretations into extensions by product sorts.

4.9.4 General interpretations

At the beginning of 4.9 we mentioned that implementation needs suggest a

variant of translation/interpretation mapping a sort to a tuple of sorts with relativisation predicate and translation of equality (Turski and Maibaum 1987, p. 265: Maibaum *et al.* 1991, p. 14). The idea of such general many-sorted interpretations is that a universe of the source language is the quotient of a part of a Cartesian product of corresponding universes of the target language. We shall now examine such general variants and see how they can be reduced to simple interpretations with introduction of appropriate sorts.

Consider many-sorted languages L' and L"; a *general translation* $I:L' \rightarrow L"$ consists, as before, of appropriate renamings, which give information concerning the underlying idea of representing a source sort as a quotient of a part of a tuple of source sorts.

- Sort renaming $I_S: S' \rightarrow S"$ assigns to each sort $s \in S'$ of L' a triple $<I_t(s), I_r(s), I_q(s)>$, with components (see figure 4.28):
  - $I_t(s)$ being a non-null sequence $<s^1,\ldots,s^k>$ of sorts of L", called the *sort-tupling* of s;
  - $I_r(s)$ being a predicate over sorts $s^1,\ldots,s^k$ of L", called the *relativisation predicate* of s;
  - $I_q(s)$ being a predicate over sorts $s^1,s^1,\ldots,s^k,s^k$ of L", called the *translation of equality* over s.
- Variable renaming $I_V: V' \rightarrow V"$ assigns to each variable $v \in V'=Var(L')$, ranging over sort s of L', a k-tuple $I_V(v)=<v^1,\ldots,v^k>$ of variables of L", with $v^1,\ldots,v^k$ ranging over, respectivley, $s^1,\ldots,s^k$, when $I_t(s)=<s^1,\ldots,s^k>$.
- Predicate renaming $I_R: R' \rightarrow R"$ assigns to each predicate symbol $r' \in R'=Prd(L')$, over sorts $s_1,\ldots,s_m$ of L', a predicate symbol $I_R(r') \in R"=Prd(L")$, over sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_m^1,\ldots,s_m^{k_m}$ of L", where $<s_1^1,\ldots,s_1^{k_1}>=I_t(s_1),\ldots,<s_m^1,\ldots,s_m^{k_m}>=I_t(s_m)$.
- Operation renaming $I_F: F' \rightarrow F"$ assigns to each operation symbol $f' \in F'=Opr(L')$, from sorts $s_1,\ldots,s_n$ to s of L', with $I_t(s)=<s^1,\ldots,s^k>$, k operation symbols $f^1,\ldots,f^k \in F"=Opr(L")$, each f^i being an operation from sorts $s_1^1,\ldots,s_1^{k_1},\ldots,s_n^1,\ldots,s_n^{k_n}$ to sort s^i, of L".

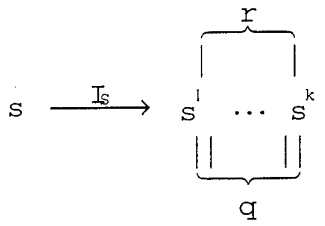

Fig. 4.28: General translation: sort renaming

These data enable translating terms and formulae form L' to L", with the same ideas as before: replacing each symbol by its translation. Since variables and operations are mapped to k-tuples, the translation of a term $t\in \mathrm{Trm}(L')$ is a sequence $I(t)=\langle t^1,\dots,t^k\rangle\in(\mathrm{Trm}(L''))^+$ of terms of L". Translation of formulae follows a similar pattern: when $I_t(s)=\langle s^1,\dots,s^k\rangle$, an equality $t\approx u$ between terms t and u of sort s in L' is translated to $q(t^1,\dots,t^k,u^1,\dots,u^k)$, an existential formula $(\exists\, v:s)\,\theta$ is translated to $(\exists v^1{:}s^1)\dots(\exists v^k{:}s^k)[r(v^1,\dots,v^k)\wedge I(\theta)]$, and a universal formula $(\forall v{:}s)\theta$ is translated to $(\forall v^1{:}s^1)\dots(\forall v^k{:}s^k)[r(v^1,\dots,v^k)\rightarrow I(\theta)]$, where $r=I_r(s)$ and $\langle v^1,\dots,v^k\rangle=I_V(v)$.

For reasons already seen, we impose some requirements for proper behaviour. For each sort $s\in \mathrm{Srt}(L')$ with $I_t(s)=\langle s^1,\dots,s^k\rangle\in(\mathrm{Srt}(L''))^k$, $I_r(s)=r\in \mathrm{Prd}(L'')[s^1,\dots,s^k]$ and $I_q(s)=q\in \mathrm{Prd}(L'')[s^1,\dots,s^k,s^1,\dots,s^k]$, these requirements are as follows.

1. Relativisation requirements:

(ν) non-voidness .requirement $\nu(r)$ for relativisation predicates:
$(\exists v^1{:}s^1)\dots(\exists v^k{:}s^k)r(v^1,\dots,v^k)$;

(χ) for each operation symbol $f\in \mathrm{Opr}(L')$, from sorts $s_1,\dots,s_n$ to s of L', with $I_F(f)=\langle f^1,\dots,f^k\rangle$, the closure requirement $\chi(r_1,\dots,r_n\rightarrow r)[f^1,\dots,f^k]$ (with implicit universal quantification):

$$[(r_1(x_1^1,\dots,x_1^{k_1})\wedge\dots\wedge r_n(x_n^1,\dots,x_n^{k_n}))\rightarrow \rightarrow r(f^1(x_1^1,\dots,x_1^{k_1},\dots,x_n^1,\dots,x_n^{k_n}),\dots,f^k(x_1^1,\dots,x_1^{k_1},\dots,x_n^1,\dots,x_n^{k_n}))].$$

2. Equality requirements:

(ε) the translations $I(\rho)$, $I(\sigma)$ and $I(\tau)$ of the sentences expressing reflexivity, symmetry and transitivity of equality $\approx$;

(f) for each operation symbol $f\in \mathrm{Opr}(L')$, from sorts $s_1,\dots,s_n$ to s of L', the congruence requirements $\kappa(I)[f{:}s_1,\dots,s_n\rightarrow s]$, i.e. the translations

$$I((\forall u_1,v_1{:}s_1)\dots(\forall u_n,v_n{:}s_n)[(u_1\approx v_1\wedge\dots\wedge u_n\approx v_n)\rightarrow \rightarrow f(u_1,\dots,u_n)\approx f(v_1,\dots,v_n)]);$$

(r) for each predicate symbol $r\in \mathrm{Prd}(L')$, over sorts $s_1,\dots,s_m$ of L', the congruence requirements $\kappa(I)[r(s_1,\dots,s_m)]$, i.e. the translations

$$I((\forall u_1,v_1{:}t_1)\dots(\forall u_m,v_m{:}t_m)[(u_1\approx v_1\wedge\dots\wedge u_m\approx v_m)\rightarrow \rightarrow(r(u_1,\dots,u_m)\rightarrow r(v_1,\dots,v_m))]).$$

Given a specification $P''=\langle L'',G''\rangle$, we say that general translation I is a *general interpretation* $I:\langle L',\varnothing\rangle\rightarrow\langle L'',G''\rangle$ iff the above requirements are in $\mathrm{Cn}[P'']$. The preceding interpretations can be considered as special cases of general interpretations when one or more parts become trivial.

4.9.5 Reduction of general to simple interpretations

We can now put together the preceding considerations to reduce a general interpretation $I_{t,r,q}:<L',\varnothing>\rightarrow<L'',G''>$ to a simple interpretation with introduction of sorts.

We shall use the sentence $(\forall v:s)(\exists w:s)v\approx w$ as an illustrative example. It can be translated by steps as follows:

(q) first to $(\forall v:s)(\exists w:s)q'(v,w)$;

(r) then to $(\forall v:s)\{r''(v)\rightarrow(\exists w:s)[r''(w)\wedge q''(v,w)]\}$,

(t) and finally to

$$(\forall v^1:s^1)\ldots(\forall v^k:s^k)\{r(v^1,\ldots,v^k)\rightarrow \rightarrow(\exists v^1:s^1)\ldots(\exists v^k:s^k)[r(w^1,\ldots,w^k)\wedge q(v^1,w^1,\ldots,v^k,w^k)]\}.$$

We then have (see figure 4.29):

(q) the first step may be regarded as a translation I_q of equality;

(r) the second step can be considered as a translation I_r with relativisaton predicate r', translating q' to q" and r' to r";

(t) the third step amounts to a translation I_t with sort tupling, translating r" and q" to r and q, respectively.

By using these translations to introduce appropriate axiomatisations, we can decompose $I_{t,r,q}$ as $I_{t,r,q}=I_q;I_r;I_t$; and we have already argued that these components are reducible to simple interpretations by properly introducing new subsorts, quotient sorts and products sorts.

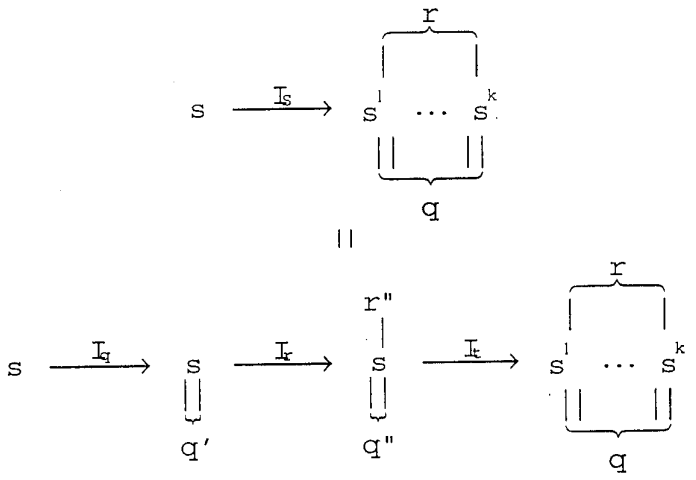

Fig. 4.29: Decomposition of general interpretation

To see an explicit reduction of a general interpretation to a simple one, consider again our example sentence σ: $(\forall v:s)(\exists w:s)v\approx w$, translated to $I_{t,r,q}(\sigma)$. We are going to construct a simple interpretation I translating σ to $I(\sigma)$: $(\forall v':s')(\exists w':s')v'\approx w'$, so that $I_{t,r,q}(\sigma)$ and $I(\sigma)$ are equivalent in the

target specification.

The idea is as follows. If $s^/$ is the quotient of sort $s^\subseteq$ by the (equivalence) predicate q^r, then $(\forall v^/:s^/)(\exists w^/:s^/)\, v^/ \approx w^/$ is equivalent to $(\forall v^\subseteq:s^\subseteq)(\exists w^\subseteq:s^\subseteq)q^r(v^\subseteq,w^\subseteq)$; the latter, if $s^\subseteq$ is the subsort of $s^\times$ relativised to predicate r^t, is equivalent to $(\forall v^\times:s^\times)\{r^t(v^\times:)\rightarrow(\exists w^\times:s^\times)[r^t(w^\times)\wedge q^t(v^\times,w^\times)]\}$, which, if $s^\times$ is the k-fold product of $s^1,\ldots,s^k$, becomes equivalent to $I_{t,r,q}(\sigma)$:

$$(\forall v^1:s^1)\ldots(\forall v^k:s^k)\{r(v^1,\ldots,v^k)\rightarrow$$
$$\rightarrow(\exists v^1:s^1)\ldots(\exists v^k:s^k)[r(w^1,\ldots,w^k)\wedge q(v^1,w^1,\ldots,v^k,w^k)]\}.$$

Here, these predicates are inherited from q and r as before.

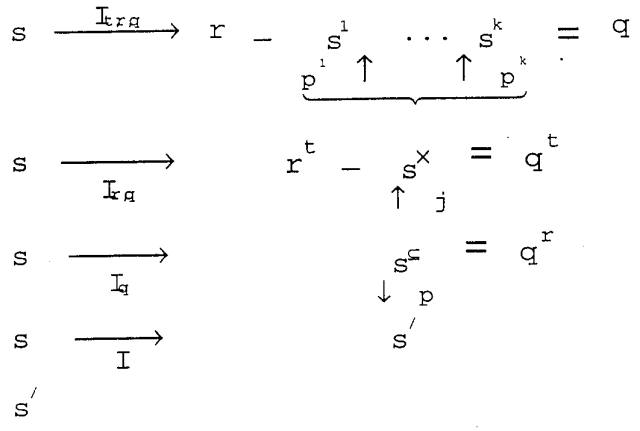

Fig. 4.30: Reducing general to simple interpretations

In other words, we proceed as follows (see figure 4.30).

(×) We first introduce $s^\times$ as the k-fold product of $s^1,\ldots,s^k$. Then, we have:

(t) predicates q^t and r^t, involving the product sort $s^\times$, induced by q and r, involving sorts $s^1,\ldots,s^k$, as in 4.9.3, where both requirements $\nu(r^t)$ and $\varepsilon(q^t)$ are guaranteed;

(r,q) a translation $I_{r,q}$ with trivial sort tupling, translating σ to $(\forall v^\times:s^\times)\{r^t(v^\times)\rightarrow(\exists w^\times:s^\times)[r^t(w^\times)\wedge q^t(v^\times,w^\times)]\}$, which is equivalent to $I_{t,r,q}(\sigma)$.

(⊆) Next, we introduce $s^\subseteq$ as the subsort of $s^\times$ relativised to predicate r^t. Then, we have:

(r) predicate q^r, over sorts $s^\subseteq,s^\subseteq$, q^t, over sorts $s^\times,s^\times$, as in 4.9.1, and the equivalence requirement $\varepsilon(q^r)$ holds;

(q) a translation I_q with sort tupling and relativisation predicate both trivial, translating σ to $I_q(\sigma)$: $(\forall v^\subseteq:s^\subseteq)(\exists w^\subseteq:s^\subseteq)q^r(v^\subseteq,w^\subseteq)$, which is equivalent to the above $I_{r,q}(\sigma)$.

(/) Finally, we introduce $s^/$ as the quotient of sort $s^\subseteq$ by the (equivalence) predicate q^r. Then, we have a translation I with sort

tupling, relativisation predicate and translation of equality all trivial, translating σ to $I(\sigma)$: $(\forall v':s')(\exists w':s')v' \approx w'$, which is equivalent to the above $I_q(\sigma)$.

Thus, a general interpretation $I_{t,r,q}$ can be reduced to a simple interpretation mapping source sort s to a quotient of a subsort of the product of $s^1,\ldots,s^k$, where $<s^1,\ldots,s^k>=I_t(s)$.

4.9.6 Reduction of many-sorted logic to unsorted logic

Many-sorted languages and specifications are quite natural and useful. But, they can - in principle - be replaced by unsorted versions without much formal loss. The reason for this is the fact that, for first-order logic, the many-sorted version can be reduced to the unsorted version (Enderton 1972, p. 279-281). We shall now examine this reduction in order to illustrate the application of some of the concepts developed in this section, mainly interpretation with relativisation.

The basic idea is very simple: given a many-sorted structure one can obtain an unsorted version whose single universe is the (disjoint) union of the universes for the sorts; given (relativisation) predicates corresponding to the sorts, the original many-sorted structure can be recovered. Thus, with (relativisation) predicates corresponding to the sorts there appears to be no loss of (model-oriented) information.

Given a many-sorted language L_+, we are going to define an unsorted language L_* and a translation $J:L_+\rightarrow L_*$. This translation - with relativisation predicates - will have some conditions $\vartheta(L_+,L_*)$ in order to ensure proprer induction of structures. Under these conditions, J will interpret each many-sorted specification $P_+=<L_+,G_+>$ faithfully into its unsorted version $P_*=<L_*,G_*\cup\vartheta(L_+,L_*)>$.

Consider a many-sorted language L_+ with set of sorts $Srt(L_+)=S_+$. We will construct its unsorted version L_*, and a translation $J:L_+\rightarrow L_*$, via a mediating single-sorted language $L_\#$ and intermediate translation $J':L_+\rightarrow L_\#$.

We construct the mediating single-sorted language $L_\#$, with $Srt(L_\#)=\{t\}$, and define $J':L_+\rightarrow L_\#$, as follows:

- each sort $s\in S_+=Srt(L_+)$ is translated to the single sort $t\in Srt(L_\#)$; $J'(s):=t$;
- for each sort $s\in S_+$, $L_\#$ has a unary predicate symbol $s^\#$, over sort t of $L_\#$, as the relativisation predicate for sort s;
- for each variable $v^+\in V^+=Var(L_+)$, over sort s of L_+, $L_\#$ has a distinct variable $v^\#:=<v^+,s>\in V^\#=Var(L_\#)$, over sort t of $L_\#$, and $J'(v^+):=v^\#$;
- for each predicate symbol $r^+\in R^+=Prd(L_+)$, over sorts $s_1,\ldots,s_m$ of L_+, $L_\#$ has

an m-ary predicate symbol $r^{\#} \in R^{\#}=Prd(L_{\#})$, over sort t of $L_{\#}$, and $J'(r^{+}):=r^{\#}$;
- for each operation symbol $f^{+} \in F^{+}=Opr(L_{+})$, from sorts $s_1,\ldots,s_n$ to s of L_{+}, $L_{\#}$ has an n-ary operation symbol $f^{\#} \in F^{\#}=Opr(L_{\#})$, over sort t of $L_{\#}$, and $J'(f^{+}):=f^{\#}$.

Now, language L_{*} is an unsorted language, which has, for each symbol $l^{\#} \in Var(L_{\#}) \cup Alph(L_{\#})$, except sort t, of $L_{\#}$ a corresponding symbol $l^{*} \in Var(L_{*}) \cup Alph(L_{*})$, of the same kind. This makes the two languages virtually identical: the translation $J^{\circ}:L_{\#} \to L_{*}$ assigning to each symbol $l^{\#} \in Var(L_{\#}) \cup Alph(L_{\#})$ its corresponding $l^{*} \in Var(L_{*}) \cup Alph(L_{*})$ is a bijective renaming. Notice that equality symbol $\approx$ of $L_{\#}$ gets translated to the unsorted equality symbol $\approx$ of L_{*}.

We now define $J:L_{+} \to L_{*}$, from many-sorted L_{+} to unsorted L_{*}, as the composite $J':L_{+} \to\!\!\!\!\!+\; L_{\#}$ followed by $J^{\circ}:L_{\#} \to L_{*}$. Notice that, except for sorts, this translation $J:L_{+} \to L_{*}$ consists of bijective renamings of symbols. In particular, each variable v^{+}, over sort s of L_{+}, is translated to a distinct variable $v^{*}:=\langle v^{+},s\rangle$ of L_{*}.

Since the intermediate translation $J':L_{+} \to L_{\#}$ has relativisation predicates, we impose some requirements for its proper behaviour (see 4.5.4.A and 4.9.1). We formulate them as their translations under $J^{\circ}:L_{\#} \to L_{*}$. These requirements form the set $\vartheta(L_{+},L_{*})$ consisting of the following sentences of L_{*}:

- for each sort $s \in Srt(L_{+})$, the non-voidness requirement $v(s^{*})$ for its relativisation predicate s^{*}: $\exists v s^{*}(v)$;
- for each operation symbol $f^{+} \in Opr(L_{+})$, from sorts $s_1,\ldots,s_n$ to s of L_{+}, the closure requirement $\chi(s^{*}_1,\ldots,s^{*}_n \to s^{*})[f^{*}]$ of relativisation predicates $s^{*}_1,\ldots,s^{*}_n$ and s^{*} with respect to f^{*}:

$$\forall x_1 \ldots \forall x_n[(s^{*}_1(x_1) \wedge \ldots \wedge s^{*}_n(x_n)) \to s^{*}(f(x_1,\ldots,x_n))].$$

By the *unsorted reduction of many-sorted language* L_{+} we mean the translation $J:L_{+} \to L_{*}$ together with the set $\vartheta(L_{+},L_{*})$ of requirements. Notice that translation $J:L_{+} \to L_{*}$ maps each formula φ^{+} of L_{+} into its translation with relativisation $J(\varphi^{+}) \in Frml(L_{*})$, as in 4.5.4.A and 4.9.1. We also have (see 4.2.2, 4.6.1, 4.5.4.A and 4.9.1) induction of structures and conjugate assignments to variables. Given a many-sorted specification $P_{+}=\langle L_{+},G_{+}\rangle$, we can use J to translate its axiomatisation G_{+} to $J(G_{+})$, and construct its *unsorted reduction* as the unsorted specification $P_{*}=\langle L_{*},G_{*}\rangle$ with

axiomatisation $G_*:=J(G_+)\cup\vartheta(L_+,L_*)$. The next result shows that this reduction behaves as it should: as a faithful interpretation.

Proposition Reduction of many-sorted to unsorted logic

Consider a many-sorted language L_+ with unsorted reduction $J:L_+\rightarrow L_*$ and $J(L_+,L_*):=\langle L^*,\vartheta(L_+,L_*)\rangle$.

a) Each model $\mathfrak{M}_*\in Mod[J(L_+,L_*)]$, with universe M_*, induces a structure $\mathfrak{M}_*\downarrow J$ for many-sorted language L_+, with Translation Connection: for each formula $\varphi^+\in Frml(L_+)$ and conjugate assignments $a_+\sim a_*$ with $a_*:Var(L_*)\rightarrow M_*$ and a_+ to $Var(L_+)$ in $\mathfrak{M}_*\downarrow J$, $\mathfrak{M}_*\downarrow J\models\varphi^+[a_+]$ iff $\mathfrak{M}_*\models J(\varphi^+)[a_*]$.

b) Given any structure $\mathfrak{M}_+$ for many-sorted language L_+, there exists a structure $\mathfrak{M}_*$ for unsorted language L_*, which is a model of $\vartheta(L_+,L_*)$ and with induced structure $\mathfrak{M}_*\downarrow J=\mathfrak{M}_+$.

c) For each many-sorted specification $P_+=\langle L_+,G_+\rangle$ with unsorted reduction $P_*=\langle L_*,G_*\cup\vartheta(L_+,L_*)\rangle$, translation $J:L_+\rightarrow L_*$

(i) interprets P_+ faithfully into its unsorted reduction P_*;

(ii) induces a correspondence from $Mod[P_*]$ onto $Mod[P_+]$.

Proof

a) The requirements in $\vartheta(L_+,L_*)$ ensure that, for each sort $s\in Srt(L_+)$, $\mathfrak{M}_+[s]:=\mathfrak{M}_*[s^*]$ is nonempty and closed under each operation $\mathfrak{M}_*[f^*]$. This providing a structure $\mathfrak{M}_*\downarrow J:=\mathfrak{M}_+$ for L_+, with $\mathfrak{M}_+[s]:=\mathfrak{M}_*[r^*]$, and $\mathfrak{M}_+[f^+]$ and $\mathfrak{M}_+[p^+]$ being the respective restrictions of $\mathfrak{M}_*[f^*]$ and $\mathfrak{M}_*[p^*]$ to the relativisation predicates corresponding to their sorts. The Translation Connection is as before.

b) Consider a structure $\mathfrak{M}_+$ for many-sorted language L_+. We construct structure $\mathfrak{M}_*$ for unsorted language L_+ as follows:

- its universe M_* is the disjoint union of the universes $\mathfrak{M}_+[s]$ for $s\in Srt(L_+)$;
- for each relativisation predicate $s^*\in I(Srt(L_+))$, $\mathfrak{M}_*[s^*]:=\mathfrak{M}_+[s]$;
- for every other predicate $r^*\in[Prd(L_*)-I(Srt(L_+))]$, $\mathfrak{M}_*[r^*]:=\mathfrak{M}_+[r^+]$;
- for each n-ary operation $f^*\in Opr(L_*)$, $\mathfrak{M}_*[f^*]$ is an extension of $\mathfrak{M}_+[f^+]$ to $(M_*)^n$.

By construction, we have a structure $\mathfrak{M}_*$ for unsorted language L_*, such that $\mathfrak{M}_*\models\vartheta(L_+,L_*)$ and with induced structure $\mathfrak{M}_*\downarrow J=\mathfrak{M}_+$.

c)Follows from (a) and (b).

QED

4.10 Examples

We now provide some examples of specifications and interpretations to illustrate some of the ideas in this section.

4.10.1 Example specifications

We first give two specifications to be used in the examples of specifications and interpretations. These are unsorted strict partial ordering and many-sorted sequences of data.

Spec. 4.1. STR_PRT_ORD: Strict partial ordering

```
SPEC STR_PRT_ORD                          {Specification of Strict partial orderings}
  DECLARATIONS
    Sorts                                                          {unsorted}
    Operations                                                     {No operations}
    Constants                                                      {No constants}
    Predicates
      ¿ bef ?                                                      {< infix binary predicate}
  AXIOMS
    ∀x¬xbefx                                                       {irreflexivity},
    ∀x,y,z[(xbefy∧ybefz)→x befz]                                   {transitivity}.
  THEOREMS                                                         {Sample consequences}
    ∀x,y(xbefy→¬ybefx])                                            {antisymmetry}.
END_SPEC STR_PRT_ORD
  many-sorted: Seq[El]
```

Spec. 4.2. SEQ[DATA]: Sequences of Data

```
SPEC SEQ[DATA]                            {Specification of Sequences of Data}
  DECLARATIONS
    Sorts
      Seq, Dt                                                      {The sorts are Seq and Dt};
    Operations
      hd (Seq)→Dt                                                  {hd gives Dt from Seq},
      tl (Seq)→Seq                                                 {tl gives Seq from Seq},
      cons (Dt,Seq)→Seq                                            {cons gives Seq from Dt & Seq},
      Seq concat Seq→Seq                                           {infix concat gives Seq from Seq & Seq};
    Constants
      nil: Seq                                                     {nil is a constant of Seq};
```

Predicates
null? (Seq) {null? is over Seq};
Variables {Declaration of variables}
q_i: Seq {Variables over sort Seq are $q_0,q_1,\ldots,q_i,\ldots$}
d_j: Dt {Variables over sort Dt are $d_0,d_1,\ldots,d_j,\ldots$}

AXIOMS
$(\forall q_0{:}Seq)\ [null?(q_0)\leftrightarrow q_0\approx nil]$ {nil vs. null?},
$(\forall q_0{:}Seq)(\forall d_0{:}Dt)\ \neg null?(cons(d_0,q_0))$ {null? vs. cons},
$tl(nil)\approx nil$ {tl of nil},
$(\forall q_0{:}Seq)(\forall d_0{:}Dt)\ tl[cons(d_0,q_0)]\approx q_0$
$(\forall q_0{:}Seq)(\forall d_0{:}Dt)\ hd[cons(d_0,q_0)]\approx d_0$
$(\forall q_1{:}Seq)\ nilconcatq_1\approx q_1$
$(\forall q_0,q_1{:}Seq)(\forall d_0{:}Dt)\ cons(d_0,q_0)concatq_1\approx cons(d_0,(q_0concatq_1)$
$Seq{:}Ind(null?(q_0);\{(\exists d_0{:}Dt)cons(d_0,q_1)\approx q_0\})$ {Seq inductive on null? & cons}

THEOREMS {Sample consequences}

COMMENTS {General remarks}
Specification of hd is underdetermined:
the value of hd(nil) is left open.

END_SPEC SEQ[DATA]

4.10.2 Example interpretations

We now give examples of translations and interpretations. We first present unsorted examples and then illustrate many-sorted cases.

A. *Unsorted interpretations*

We use unsorted Strict partial ordering to illustrate simple translation and formula translation for predicate.

A simple translation I from the language of Strict partial ordering (in Spec. 4.1) to that Integers (see Spec 2.5 in 2.10) can be presented as follows.

Transl. 4.1. From Strict partial ordering to Integers

TRNSL STR_PRT_ORD→INT {Translation from STR_PRT_ORD into INT}
Sorts {unsorted}
Operations {No operations}
Constants {No constants}
Predicates
¿ bef ? → < {bef translated to <}

END_TRNSL STR_PRT_ORD→INT

Notice that I is an injective, but not surjective, mapping.

The translations of the axioms of STR_PRT_ORD are:

$I(\forall x \neg x \underline{bef} x) = \forall x \neg x < x$ {a theorem of INT},

$I(\forall x,y,z[(x \underline{bef} y \wedge y \underline{bef} z) \rightarrow x \underline{bef} z]) = \forall x,y,z[(x<y \wedge y<z) \rightarrow x<z]$ {an axiom of INT}.

Thus, the above translation I is an interpretation from STR_PRT_ORD to INT. It is not faithful, because the ordering of the integers is linear, which the original partial ordering is not required to be: the sentence $\forall x,y[\neg x \underline{bef} y \rightarrow (x \approx y \vee y \underline{bef} x)]$ is not a theorem of STR_PRT_ORD but its translation is a theorem of INT.

A formula translation for predicate J from the language of Strict partial ordering (in Spec. 4.1) to that of Integer Arithmetic (see Spec 2.6 in 2.10) can be obtained by translating binary predicate $\underline{bef}$ of STR_PRT_ORD to the formula $\neg x \approx y \wedge \exists w\ x * w \approx y$.

B. *Many-sorted interpretations*

We use many-sorted Sequences of Data, and some variants, to illustrate simple translation as well as translation with relativisation and translation of equality.

A translation I from the language of Sets of Elements (see Spec 2.4 in 2.10) to that Sequences of Data (in Spec 4.2) can be presented as follows.

Transl. 4.2. From Stacks of Elements to Sequences of Data

TRNSL STACK[ELEMENT]→SEQ[DATA]

			{Translation of sorts}
Sorts			
Stk	→	Seq	{Set translated to Seq}
Elm	→	Dt	{Elm translated to Dt}
			{Translation of operations}
Operations			
push	→	cons	
pop	→	tl	
top	→	hd	
			{Translation of constants}
Constants			
crt	→	nil	{crt:Stk translated to nil:Seq}
			{Translation of predicates}
Predicates			
is_null?	→	null?	
			{Translation of variables}
Variables			
s_i	→	q_i	{s_i:Stk translated to q_i:Seq}
e_j	→	d_j	{e_j:Elm translated to d_j:Dt}

END_TRNSL STACK[ELEMENT]→SEQ[DATA]

The translations of the non-schema axioms of STACK[ELEMENT] are (with .plicit universal quantification):

$I[\neg crt \approx push(s_0,e_0)] = \neg nil \approx cons(d_0,q_0)$ {a theorem of SEQ[DATA]},

$I[pop(push(s_0,e_0))\approx s_0]=[tl(cons(d_0,q_0))\approx q_0]$ {an axiom of SEQ[DATA]},

$I[top(push(s_0,e_0))\approx s_0]=[hd(cons(d_0,q_0))\approx d_0]$ {an axiom of SEQ[DATA]},

$I[is\_null?(s_0)\leftrightarrow s_0\approx crt]=[null?(q_0)\leftrightarrow q_0\approx nil]$ {an axiom of SEQ[DATA]},

The inductive schema $Ind(s_0\approx crt;\{(\exists e_0{:}Elm)push(s_1,e_0)\approx s_0\})$ is equivalent to $\{\varphi(crt)\wedge(\forall s_1{:}Stk)[\varphi(s_1)\rightarrow(\forall e_0{:}Elm)\varphi(push(s_1,e_0))]\}\rightarrow(\forall s_0{:}Stk)\varphi(s_0)]$ (see 2.5). The translation of the latter, letting $I(\varphi(s_0)):=\psi(q_0)$, is $\{\psi(nil)\wedge(\forall q_1{:}Seq)[\psi(q_1)\rightarrow(\forall d_0{:}Dt)\psi(cons(d_0,q_1))]\}\rightarrow(\forall q_0{:}Seq)\psi(q_0)]$, which is equivalent to the inductive schema $Ind(null?(q_0);\{(\exists d_0{:}Dt)cons(d_0,q_1)\approx q_0\})$.

Thus, the above translation I is an interpretation from STACK[ELEMENT] to SEQ[DATA]. It is not faithful, because pop(crt) is left open in STACK[ELEMENT], but this does not happen with tl(nil) in SEQ[DATA]: the sentence pop(crt)≈crt is not a theorem of STACK[ELEMENT] but its translation is a theorem of SEQ[DATA], in fact axiom tl(nil)≈nil.

The next example involves a sub-language for sets. Given SET[ELEMENT] of Sets of Elements (see Spec. 2.4 in 2.10), consider its sub-language SMPL_ST[ELEMENT] obtained by removing operations ins and rem, and call Trv(SMPL_ST[ELEMENT]) its trivial specification.

Consider again Spec 4.2 of Sequences of Data. We can extend it to a specification SEQ[DATA]withIS_IN by introducing a binary predicate is_in between sorts Dt and Seq as well as a new axiom $(\forall d_0{:}Dt)(\forall q_0{:}Seq)\ \{d_0 is\_in q_0\leftrightarrow[(\neg null?(q_0)\wedge hd(q_0)\approx d_0)\vee d_0 is\_intl(q_0)]\}$.

For an example of translation with subsort, we extend SEQ[DATA]withIS_IN further to SEQ[DATA]withNO_REPT by introducing unary predicate no_rept over sort Seq via the defining axiom

$(\forall q_0{:}Seq)\ \{no\_rept(q_0)\leftrightarrow(\forall q_1,q_2{:}Seq)\ [\ q_1 concat q_2\approx q_0\rightarrow$
$\rightarrow(\forall d_0{:}Dt)\ (d_0 is\_in q_1\rightarrow\neg d_0 is\_in q_1)]$.

The non-voidness requirement for no_rept is guaranteed by no_rept(nil).

We can then assign to sort Set both sort Seq and relativisation predicate no_rept over sort Seq, and define an interpretation with relativisation predicates of Trv(SMPL_ST[ELEMENT]) into SEQ[DATA]withNO_REPT as follows.

Transl. 4.3. From Simple Sets of Elements to Sequences of Data

INTRPRT_RELTV Trv(SMPL_ST[ELEMENT])→SEQ[DATA]withNO_REPT

Sorts			{Translation of sort to sort;predicate}
Set	→	Seq;no_rept	
Elm	→	$Dt;d_0\approx d_0$	{trivial relatvisation}
Operations			{Translation of operations}
sel	→	hd	{sel translated to hd}
Constants			{Translation of constants}

void	→	nil	
Predicates			{Translation of predicates}
empty?	→	null?	
blng	→	is_in	{prefix translated to infix}
Variables			{Translation of variables}
t_i	→	q_i	{t_i:Set translated to q_i:Seq}
e_j	→	d_j	{e_j:Elm translated to d_j:Dt}

END_INTRPRT_RELTV TRV(SMPL_ST[ELEMENT])→SEQ[DATA]withNO_REPT

Notice that sentence $(\forall t_0{:}Set)[\neg empty?(t_0)\rightarrow blng(sel(t_0),t_0)]$ of language SMPL_ST[ELEMENT] - which is an axiom of SET[ELEMENT] - is translated to $(\forall q_0{:}Seq)\{no\_rept(q_0)\rightarrow[\neg null?(q_0)\rightarrow hd(q_0)is\_inq_0]\}$.

The next example involves the sub-language QRY_ST[ELEMENT] of the language SMPL_ST[ELEMENT] obtained by further removing operation sel. Call Trv(QRY_ST[ELEMENT]) its trivial specification.

For an example of translation of equality, we extend SEQ[DATA]withIS_IN to SEQ[DATA]withSAME by defining binary predicate same over sort Seq via the $(\forall q_0,q_1{:}Seq)$ $[same(q_0,q_1)\leftrightarrow(\forall d_0{:}Dt)(d_0is\_inq_1\leftrightarrow d_0is\_inq_1)]$.
The equivalence requirement for same is clearly guaranteed.

We can then assign to sort Set both sort Seq and equivalence predicate same over sort Seq, and define an interpretation with translation of equality of Trv(QRY_ST[ELEMENT]) into SEQ[DATA]withSAME as follows.

Transl. 4.4. From Simple Sets of Elements to Sequences of Data

INTRPRT_EQLT Trv(QRY_ST[ELEMENT])→SEQ[DATA]withSAME

Sorts			{Translation of sort to sort/equivalence}
Set	→	Seq/same	,
Elm	→	$Dt/d_0{\approx}d_1$	{trivial equivalence};
Operations			{no operations}
Constants			{Translation of constants}
void	→	nil	
Predicates			{Translation of predicates}
empty?	→	null?	
blng	→	is_in	{prefix translated to infix}
Variables			{Translation of variables}
t_i	→	q_i	{t_i:Set translated to q_i:Seq}
e_j	→	d_j	{e_j:Elm translated to d_j:Dt}

END_INTRPRT_EQLT TRV(QRY_ST[ELEMENT])→SEQ[DATA]withSAME

Now the sentence $(\forall t_0,t_1{:}Set)[t_0{\approx}t_1\leftrightarrow(\forall e_0{:}Elm)(blng(e_0,t_0)\leftrightarrow blng(e_0,t_1))]$ of language QRY_ST[ELEMENT] - which is an axiom of SET[ELEMENT] - is

translated to the axiom defining predicate same.

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 (1984) A systematic study of models of abstract data types. *Theor. Comp. Sci..*, **33** 139-174. {Preliminary version: Centre de Recherche en Informatique de Nancy Res. Rept. 81-R-042, Nancy.}

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** (1) 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. and Meseguer, J. (1981) Completeness of many-sorted equational logic. *ACM Sigplan Notices* **16** (7) 24-32.

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.

Meré, M. C. ; Veloso, P. A. S. (1995) Definition-like extensions by sorts. *Bull. IGPL*, **5** (4), 579-595.

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. *Bull. 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.

Zilles, S. N. (1974) Algebraic specification of abstract data types. Computation Structures Group Memo 119, Lab. for Computer Science, MIT, Cambridge.