Preservation and Reflection in Specification(*)

A.Lopes and J.L.Fiadeiro

Department of Informatics, Faculty of Sciences, University of Lisbon
Campo Grande, 1700 Lisboa, PORTUGAL
{mal,llf}@di.fc.ul.pt

Abstract. We extend the traditional notion of specification based on theories and interpretations between theories to model situations, typical of open, reactive systems, in which properties exhibited locally by an object no longer hold when that object is interconnected as a component of a larger system. The proposed notion of specification is based on the observation, due to Winskel, that while some assertions are *preserved* across morphisms of labelled transition systems, other are *reflected*. The distinction between these two classes of assertions leads us to the definition of two categories of specifications, one that supports horizontal structuring and another that supports vertical structuring, for which compositionality is proved.

1 Introduction

The notion of a specification as a collection of sentences in some logic (theory presentation) expressing the properties that the program is required to satisfy, and of specification morphism as a property preserving mapping (interpretation between theories), has served an important rôle in supporting software development, namely the process of stepwise refinement of high level descriptions of the functionality of the system [15]. These notions have also been useful for structuring specifications within a given layer of abstraction [4], giving rise to algebraic frameworks where both dimensions – *horizontal* (for structuring) and *vertical* (for refinement) – can be integrated [12].

Although these techniques were developed for supporting Abstract Data Type specification, i.e. for supporting the development of transformational programs, they seem to serve other programming paradigms namely those which, like object-oriented programming, adopt a "design-by-contract" style of development [17]. The contract (specification) consists of properties that the user, or client, of an object can rely on because any implementation is required to satisfy them.

However, when it comes to supporting the process of horizontal structuring in the context of reactive systems, i.e. of interconnecting objects to form a (complex) system, it seems that property preservation is not the right structural notion on which an algebraic approach can be based. For instance, typical clauses in an OO-contract are conditions under which methods are guaranteed to be available (pre-conditions in the sense of OO [17]). It is easy to understand that, in a synchronous mode of interconnection in which objects interact by sharing actions, the availability of an action (method) is determined by the intersection

(*) This work was partially supported by the project ARTS under contract to Equitel S.A., and through the contracts PRAXIS XXI 2/2.1/MAT/46/94 (ESCOLA) and PCSH/OGE/1038/95 (MAGO).

of the availability conditions of all the objects that synchronise in that action. Hence, the property of availability (readiness) is not preserved when moving from an individual object to a system.

The fact that certain properties do not carry forwards from components to systems is well known in open, reactive system development. The difficulty is, usually, to develop a specification formalism that is *compositional* with respect to a given program design formalism and, at the same time, accounts for all relevant classes of properties (e.g., [20, 14]). The "guilty" party in this context is usually the class of liveness properties (e.g., [2]). In the context of an algebraic approach to specification [8], this means that a functorial relationship between a category of programs (or lower-level specifications) and a category of specifications cannot be established [6].

In order to account for these phenomena, other styles of specification have been developed, mainly in the context of state-based approaches, which relativise the commitment to guarantee certain properties to certain assumptions on the behaviour of the environment [1, 5]. The crucial difference here with respect to "older" styles is the fact that the interference of an environment is taken into account in the specification.

Our aim in this paper is to incorporate these notions of interference of an environment in the algebraic approach to specification based on institutions [13]. Our approach is based on the fact, also observed in [19] in the context of local logics of labelled transition systems, that, whereas certain properties are *preserved* across morphisms, other can only be *reflected* in the sense that they will only be ensured as long as the environment of the component co-operates.

More specifically, in section 2, we propose that specifications include two sets of sentences: one expressing the properties that are ensured in any context, and the other expressing the properties that a component is willing to have when working as a component of a larger system, i.e. describing the degree of co-operation towards the environment. We call the latter *co-properties*.

In section 3, we formalise the difference between the rôles of these two sets of sentences by defining two different categories of specifications. One captures the component-of relationship, and supports horizontal structuring. Morphisms of this category preserve properties and reflect co-properties. The finite co-completeness of this category is proved and pushouts are characterised. The other category captures refinement (vertical structuring) by having morphisms that preserve both properties and co-properties, thus coinciding with the traditional specification morphisms as interpretations between theories. We then prove that the use of distinct morphisms for supporting horizontal and vertical composition is compositional.

2 Specification of Reactive Systems

2.1 The Specification Logic

The logic that we will use to illustrate the proposed revision of the notion of

specification and specification morphism is a modal action logic (MAL) similar to the one that was developed in the context of the FOREST project [16] and to the basic propositional modal logics described in [18]. Notice that this use of modal logic is different from local logics of labelled transition systems in the style of Hennessy-Milner as used, for instance, in [19]. We present MAL in the style of institutions [13].

We denote by *SET* the category of sets and total functions and by $SET_\perp$ the category of pointed sets: the objects are pairs $<A,\perp>$ where A is a set and $\perp$ is a distinguished element of A; a morphism σ from $<A',\perp'>$ to $<A,\perp>$ is a mapping $\sigma:A'\rightarrow A$ s.t. $\sigma(\perp')=\perp$.

Definition/Proposition 2.1. Signatures are pairs $<A,\Gamma>$ where A is a finite set and Γ is a finite pointed set. A signature morphism σ from $\Sigma=<A,\Gamma>$ to $\Sigma'=<A',\Gamma'>$ is a pair $<\sigma_{at}:A\rightarrow A',\sigma_{ac}:\Gamma'\rightarrow\Gamma>$ where σ_{at} is a mapping and σ_{ac} is a morphism in $SET_\perp$. Signatures and signature morphisms constitute a category *SIGN*. ∎

As in [8], a signature $<A,\Gamma>$ provides the (non-logical) symbols of the language used for specifying a system: *A* is the set of attribute symbols and Γ is the set of action symbols. In order to simplify the presentation we only consider Boolean attributes. As in [7], the distinguished action $\perp$ of Γ represents idle steps, that is, the steps performed by the environment, reflecting the fact that we are considering systems embedded in a wider environment.

A signature morphism σ from Σ to Σ' is intended to support the identification of a way in which a system (with signature Σ) is a component of, or is refined by, another system (with signature Σ'). Hence, such a morphism provides, for each attribute $a\in\Sigma$ of the component, the corresponding attribute $\sigma(a)\in\Sigma'$ of the system, and identifies the action of the component which is involved, or refined by, each action of the system, if ever. Notice that the condition $\sigma(\perp')=\perp$ states that the environment of Σ' is part of the environment of Σ. On the other hand, actions $a'\in\Sigma'$ of the system may be mapped to an idle step of the component – $\sigma(a')=\perp$ – meaning that σ identifies part of the environment of the component.

We shall see in section 3 at what level the crucial difference between refinement and composition is established. From the previous paragraph it is obvious that this distinction cannot be made at the signature level but at a more "semantic" level which considers the way systems behave.

Definition/Proposition 2.2. The grammar functor $MAL:SIGN\rightarrow SET$ defines, for every signature $\Sigma=<A,\Gamma>$, the set of modal propositions $MAL(\Sigma)$ as follows:

$$\phi ::= a \mid (\neg\phi) \mid (\phi\supset\phi') \mid \mathbf{beg} \mid [\gamma]\phi$$

for $a\in A$ and $\gamma\subseteq\Gamma$. A signature morphism $\sigma=<\sigma_{at},\sigma_{ac}>:\Sigma\rightarrow\Sigma'$ induces a translation $MAL(\sigma):MAL(\Sigma)\rightarrow MAL(\Sigma')$, which we denote by $\underline{\sigma}$, defined as follows:

$$\underline{\sigma}(\phi) ::= \sigma_{at}(a) \mid (\neg\underline{\sigma}(\phi)) \mid (\underline{\sigma}(\phi)\supset\underline{\sigma}(\phi')) \mid \mathbf{beg} \mid [\sigma_{ac}^{-1}(\gamma)]\underline{\sigma}(\phi)$$ ∎

For each $\gamma\subseteq\Gamma$, the modal operator $[\gamma]$ is such that the formula $[\gamma]\phi$ expresses that, for every g in γ, ϕ holds after *g* occurs. The propositional constant **beg** denotes the initial state. (See the semantics below.)

We will also work with some derived operators. In particular, for each $\gamma\subseteq\Gamma$, the dual of $[\gamma]$ is the modal operator $<\gamma>$ defined by the abbreviation $<\gamma>\phi\equiv_{abv}$

$(\neg[\gamma](\neg\phi))$. The formula $<\gamma>\phi$ expresses that there exists an action g in γ that establishes ϕ. We also adopt the abbreviation of a singleton by its element, that is, for every $g\in\Gamma$, $[g]\phi \equiv_{abv} [\{g\}]\phi$ and $<g>\phi \equiv_{abv} <\{g\}>\phi$.

The translation of formulas induced by a translation of the non-logical symbols was defined inductively, in the usual way. Notice, however, the use of the pre-image in the translation of the modality: properties asserted on actions of the component are translated into properties of every action of the system that involves the component actions. As in [18], this modal language is interpreted on labelled transition systems.

Definition/Proposition 2.3. The model functor $MOD:SIGN\rightarrow SET^{op}$ is defined as follows: for every signature $\Sigma=<A,\Gamma>$, a Σ-model consists of a quadruple $<W,R,V,w_0>$ where W is a set, $R:\Gamma\rightarrow[W\rightarrow 2^W]$ and $V:A\rightarrow 2^W$ are mappings, and $w_0\in W$, s.t., for every $w\in W$, $R(\perp)(w)\neq\emptyset$. We denote by $MOD(\Sigma)$ the set of all Σ-models. A Σ-model $M=<W,R,V,w_0>$ is called a locus iff, for every $w\in W$ and $w'\in R(\perp)(w)$, $V(w')=V(w)$. Given a signature morphism $\sigma=<\sigma_{at},\sigma_{ac}>:\Sigma\rightarrow\Sigma'$, for every Σ-model $M=<W,R,V,w_0>$, its reduct $MOD(\sigma)(M)$, which we denote by $M_{|\sigma}$, is defined by $<W,R_{|\sigma},V_{|\sigma},w_0>$ where $R_{|\sigma}(g)(w)=\bigcup_{\sigma(g')=g} R(g')(w)$ and $V_{|\sigma}(a)=V(\sigma_{at}(a))$. ∎

In a Σ-model $<W,R,V,w_0>$, W is the set of states, R provides, for every action, a transition relation, V is a valuation function providing, for each attribute, the set of states in which it holds, and w_0 is the initial state. We will also denote by V its dual, i.e. the mapping $W\rightarrow 2^A$ that returns the set of the atributes that are true at each state.

Recalling that $\perp$ denotes an environment step, the condition $R(\perp)(w)\neq\emptyset$ means that a system cannot prevent the environment from progressing on its own. This means that we are working with an *open* semantics of behaviour [3][1]. Moreover, the condition $V(w')=V(w)$, for every $w'\in R(\perp)(w)$, means that a locus is a model in which the attributes remain unchanged whenever the component remains idle, i.e. attributes are local (encapsulation) [8].

Definition 2.4. For every signature $\Sigma=<A,\Gamma>$, the satisfaction relation is defined as follows: a Σ-proposition ϕ is said to be true in a Σ-model $M=<W,R,V,w_0>$ at state w (which we write $(M,w)\models_\Sigma\phi$) iff:

- $(M,w)\models_\Sigma a$ iff $w\in V(a)$;
- $(M,w)\models_\Sigma\neg\phi$ iff $(M,w)\not\models_\Sigma\phi$;
- $(M,w)\models_\Sigma\phi\supset\phi'$ iff $(M,w)\models_\Sigma\phi$ implies $(M,w)\models_\Sigma\phi'$;
- $(M,w)\models_\Sigma$ **beg** iff $w=w_0$;
- $(M,w)\models_\Sigma[\gamma]\phi$ iff for every $g\in\gamma$, for every $w'\in R(g)(w)$, $(M,w')\models_\Sigma\phi$.

A Σ-proposition ϕ is said to be true in M, which we denote by $M\models_\Sigma\phi$, iff $(M,w)\models_\Sigma\phi$ for every $w\in W$. A Σ-proposition ϕ is said to be valid, which we denote by $\models_\Sigma\phi$, iff $M\models_\Sigma\phi$ for every locus M. ∎

Note that, by the notion of validity defined above, a valid formula is expected to hold at all states but only for the models which are loci. We consider that:

- $\Phi\models_\Sigma\varphi$ iff, for every Σ-locus M, if $M\models_\Sigma\phi$ for every $\phi\in\Phi$, then $M\models_\Sigma\varphi$;

---

1 A similar property is used in state-based approaches, e.g. [1], for characterising open semantics.

- $\Phi \models_\Sigma \Psi$ iff, for every $\psi \in \Psi$, $\Phi \models_\Sigma \psi$;
- $\Phi^\bullet = \{\phi \in MAL(\Sigma): \Phi \models_\Sigma \phi\}$.

The definition of validity based only on the models which are loci gives rise to a logic for which the satisfaction condition of institutions does not hold. Following [10], in this case we obtain a "weakly-structural" logic, i.e., consequence is not preserved by translation, it depends on the the locality conditions which characterise the models which are loci.

Proposition 2.5. $\langle SIGN, MAL, \models \rangle$ is a weakly structural π-institution, i.e., given a signature morphism $\sigma:\Sigma \to \Sigma'$:

(1) for every $\phi \in MAL(\Sigma)$ and $\Phi \subseteq MAL(\Sigma)$, if $\Phi \models_\Sigma \phi$ then $\underline{\sigma}(\Phi), \underline{\sigma}(loc(\Sigma)) \models_{\Sigma'} \underline{\sigma}(\phi)$,

(2) $\models_\Sigma \phi$ for every $\phi \in loc(\Sigma)$,

(3) $\underline{\mu}(\underline{\sigma}(loc(\Sigma)), \underline{\mu}(loc(\Sigma')) \models_{\Sigma''} \phi$ for every $\phi \in \underline{\sigma;\mu}(loc(\Sigma))$,

(4) $\underline{\sigma;\mu}(loc(\Sigma)) \models_{\Sigma''} \phi$ for every $\phi \in \underline{\mu}(\underline{\sigma}(loc(\Sigma)))$,

where, given a signature $\Sigma = \langle A, \Gamma \rangle$, $loc(\Sigma) = \{a \supset [\bot]a, \neg a \supset [\bot]\neg a : a \in A\}$. ∎

2.2 Specifications

As already motivated in the introduction, we claim that open, reactive systems, should be described not only in terms of the properties that they ensure in any context but also of the properties that they are willing to have when working as a component of a larger system. We shall call the latter *co-properties* in the sense that they *reflect* the behaviour of the environment rather than induce behaviour on the environment. The use of the prefix *co* is also meant to suggest that these properties establish the degree of *co*-operation of a component in regard to its environment. Naturally, co-properties will be observed when the system runs in isolation but it is possible that they are not observed when the system is interconnected to other components in a larger system.

The nature of properties and co-properties is determined by the computation and interconnection models that are adopted: it is the discipline of component interconnection available at program design level that determines which formulas express properties, in the sense that they are preserved, and which formulas express co-properties, in the sense that they are reflected. In the specific case that we are using for illustration, for which interconnection is based on local, private attributes and global, shared actions, properties address the functionality ("partial correctness") of the transition system (invariants, pre-conditions, effects of the actions over the attributes) whereas co-properties are concerned with the readiness to perform certain actions (enabling conditions). The fact that the former are preserved results from locality of attributes. The fact that the latter cannot be preserved results from the fact that actions can be shared. Indeed, co-properties typically require the cooperation of the other components (environment) to be observed. This is, for instance, the case of readiness properties: a system is ready to execute an action only if the other components that are involved in the execution of that action are ready to execute it.

Properties and co-properties are supported by two sublanguages:

Definition 2.6. The mapping *STAT*: | *SET* |→| *SET* | defines, for every set A, the set of state propositions *STAT*(A) as follows:

$$\varphi \;:=\; a \mid (\neg\varphi) \mid (\varphi \supset \varphi')$$

for $a \in A$. The mapping *PROP*: | *SIGN* |→| *SET* | defines, for every signature $\Sigma=\langle A,\Gamma\rangle$, the set of properties *PROP*(Σ) as follows:

$$\phi \;:=\; \varphi \mid (\mathbf{beg} \supset \varphi) \mid (\varphi \supset [\gamma]\varphi')$$

for $\gamma \subseteq \Gamma$ and $\varphi,\varphi' \in$ *STAT*(A). The mapping *co-PROP*: | *SIGN* |→| *SET* | defines, for every signature $\Sigma=\langle A,\Gamma\rangle$, the set of co-properties *co-PROP*(Σ) as follows:

$$\psi \;::=\; \varphi \mid (\varphi \supset \langle g\rangle \mathbf{true})$$

for $g \in \Gamma$ and $\varphi \in$ *STAT*(A). ∎

As already motivated, properties capture invariants (φ), initialisation conditions ($\mathbf{beg} \supset \varphi$), effects of actions ($\varphi \supset [g]\varphi'$), and restrictions to the occurrence of actions ($\varphi \supset [g]\mathbf{false}$). Co-properties capture the ability of actions to occur in certain states – readiness ($\varphi \supset \langle g\rangle \mathbf{true}$) – and also state propositions (φ). That is, state propositions can be used both as properties (guaranteeing that they will hold in any context) and as co-properties (expressing the fact that the component will reflect them).

Notice that formulas of type ($\varphi \supset \langle\gamma\rangle \mathbf{true}$), where γ is not a singleton, express internal nondeterminism and should not be considered as properties (they cannot be imposed on the environment) nor co-properties (they cannot be reflected because refinement is not required to preserve internal non-determinism – see section 3.3).

Definition 2.7. A specification is a triple $\langle\Sigma,\Phi,\Psi\rangle$, where Σ is a signature in *SIGN*, $\Phi \subseteq$ *PROP*(Σ) and $\Psi \subseteq$ *co-PROP*(Σ). ∎

As an example, consider a vending machine that, once it accepts a coin, is ready to serve either a cake or a cigar, according to the selection made by its user; after delivering a cake or a cigar it only accepts a new coin; and is initialised so as to accept only coins. This object can be specified as follows.

specification vending machine **is**

attributes	ready		
actions	coin, cake, cigar		
axioms	**beg** ⊃ ¬ready	*co-axioms*	ready ⊃ <cake>**true**
	ready ⊃ [coin]**false**		ready ⊃ <cigar>**true**
	¬ready ⊃ [cake,cigar]**false**		
	[coin]ready		
	[cake,cigar]¬ready		

The absence of a co-axiom expressing the conditions under which *coin* is enabled implies that, whenever the machine is not ready, it can either accept or refuse a coin, depending on the result of an internal choice. As we shall see in section 3.3, this form of (allowed) non-determinism, modelled by underspecification, can be restricted during a refinement step, i.e. a choice can be imposed as a result of a refinement step. However, the property that the selection between delivering a cake or a cigar should be made by the user amounts to required non-determinism that is materialised in the specification through the two readiness (co-)axioms. These are properties that any refinement has to enforce.

In section 3, we will illustrate how the co-axioms of a component may not be preserved in the system, and how the co-axioms of the system are reflected in every component.

It is important to mention that the translations induced by signature morphisms preserve the category of state propositions and properties, that is:

Proposition 2.8. The mappings *STAT* and *PROP* defined in 2.6 extend to functors, considering that the translation induced by a signature morphism σ coincides with the translation $MAL(\sigma)$ of these formulas. ∎

This does not hold, however, for co-properties because of the pre-image taken over the action that occurs in the modality: $(\underline{\sigma}(\varphi) \supset \langle \sigma_{ac}^{-1}(g) \rangle \underline{\sigma}(\varphi'))$ may not be a co-property because $\sigma_{ac}^{-1}(g)$ is not necessarily a singleton in which case, as already mentioned, the formula expresses internal non-determinism, which cannot be always reflected.

3 Categorical Approach to Specification

The formal justification for changing the notion of specification from theory presentations, as usual, to pairs of presentations over the same signature has to come from the fact that axioms and co-axioms behave differently when specification morphisms are considered. This was, indeed, the motivation for the proposed revision of the notion of specification: during horizontal composition, i.e. while structuring a complex system by interconnecting simpler components, axioms are preserved but co-axioms are reflected. We start by formalising and illustrating this behaviour through a new notion of specification morphism.

We then investigate how these new morphisms interact with the traditional notion of interpretation between theories (which supports refinement), and prove that the development framework that results from the use of distinct morphisms for supporting horizontal and vertical composition is compositional, meaning that, in this framework, systems can be specified independently of their environment and, since there is an explicit mechanism for putting systems together, they can be used (and reused) in several ways as components of larger systems.

3.1 The Component-of Relation

We investigate the kind of relationship that must exist between the specifications of two systems so that one can be considered as a component of the other.

As mentioned in the previous section, a specification $\langle \Sigma, \Phi, \Psi \rangle$ of a system S states that S guarantees to satisfy the set of properties $\Phi^{\bullet}$ in any context. More concretely, when S is working as a component of a system S', via a signature morphism σ from Σ to Σ', S guarantees to satisfy the properties $\underline{\sigma}(\Phi^{\bullet})$ and, therefore, the properties that S' guarantees to satisfy include $\underline{\sigma}(\Phi^{\bullet})$. In a weakly structural π-institution this corresponds to $\Phi' \models_{\Sigma'} \underline{\sigma}(\Phi)$ and $\Phi' \models_{\Sigma'} \underline{\sigma}(\mathrm{loc}(\Sigma))$.

On the other hand, the set of co-axioms Ψ constrains the co-properties that a system S' may be willing to have in any context, given that S' includes S as a

component. More specifically, for a system to be ready to perform an action *g* it is required that each of its components be ready to perform *the action in g* (if any) in which that component is involved (given by $\sigma_{ac}(g)$ for a component identified via σ). Because, as remarked after proposition 2.8, co-properties do not translate to co-properties, it seems clear that we need another way of accounting for the reflection, namely a different translation for co-properties. More precisely, we are going to define, for every signature morphism σ, a mapping $\overline{\sigma}$ (co-translation) such that, for any set Ψ of co-properties, $\overline{\sigma}(\Psi)$ is the set of co-properties that a system which includes S as a component via σ can be willing to have given that S is willing to have Ψ.

Just like the language of properties for a signature Σ is a subset of the language $MAL(\Sigma)$, for which the morphisms satisfy the structural results typical of institutions (e.g. 2.5), the language of co-properties defined in 2.6 and the required co-translation mechanisms can be put in the context of a more general grammar – *co-MAL*.

Definition/Proposition 3.1. The grammar functor *co-MAL*:*SIGN*$\rightarrow$*SET* defines, for every signature $\Sigma=\langle A,\Gamma\rangle$, the set of modal co-propositions *co-MAL*(Σ) as follows:

$$\phi ::= a \mid (\neg\phi) \mid (\phi\supset\phi') \mid \mathbf{beg} \mid \langle[\gamma]\rangle\phi$$

for $a\in A$, $\gamma\subseteq\Gamma$. A signature morphism $\sigma=\langle\sigma_{at},\sigma_{ac}\rangle:\Sigma\rightarrow\Sigma'$ induces a (co)translation *co-MAL*(σ):*co-MAL*(Σ)$\rightarrow$*co-MAL*(Σ'), which we denote by $\overline{\sigma}$, defined as follows:

$$\overline{\sigma}(\phi) ::= \sigma_{at}(a) \mid (\neg\overline{\sigma}(\phi)) \mid (\overline{\sigma}(\phi)\supset\overline{\sigma}(\phi')) \mid \mathbf{beg} \mid \langle[\sigma_{ac}^{-1}(\gamma)]\rangle\overline{\sigma}(\phi)$$ ∎

For each $\gamma\subseteq\Gamma$, the modal operator $\langle[\gamma]\rangle$ is such that the formula $\langle[\gamma]\rangle\phi$ expresses that, for some g in γ, ϕ holds after *g* occurs.

As before, we will also work with the dual operators. In particular, for each $\gamma\subseteq\Gamma$, the dual of $\langle[\gamma]\rangle$ is the modal operator $[\langle\gamma\rangle]$ defined by the abbreviation $[\langle\gamma\rangle]\phi \equiv_{abv} (\neg\langle[\gamma]\rangle(\neg\phi))$. We also adopt the abbreviation $[g]\phi \equiv_{abv} \langle[\{g\}]\rangle\phi$ and $\langle g\rangle\phi \equiv_{abv} [\langle\{g\}\rangle]\phi$, for every $g\in\Gamma$.

At the level of models, the reflection of co-properties is captured in the reverse direction through an expansion of component models into system models (recall that preservation is captured through reducts of system models into component models). Due to space limitations, details will be presented elsewhere.

Definition 3.2. For every signature $\Sigma=\langle A,\Gamma\rangle$, the satisfaction relation is defined as follows: a Σ-co-proposition ϕ is said to be true in a Σ-model $M=\langle W,R,V,w_0\rangle$ at state w (which we write $(M,w)\models_\Sigma\phi$) iff:

- $(M,w)\models_\Sigma a$ iff $w\in V(a)$;
- $(M,w)\models_\Sigma\neg\phi$ iff $(M,w)\not\models_\Sigma\phi$;
- $(M,w)\models_\Sigma\phi\supset\phi'$ iff $(M,w)\models_\Sigma\phi$ implies $(M,w)\models_\Sigma\phi'$;
- $(M,w)\models_\Sigma$ **beg** iff $w=w_0$;
- $(M,w)\models_\Sigma\langle[\gamma]\rangle\phi$ iff for some $g\in\gamma$, for every $w'\in R(g)(w)$, $(M,w')\models_\Sigma\phi$.

A Σ-co-proposition ϕ is said to be true in M, which we denote by $M\models_\Sigma\phi$, iff $(M,w)\models_\Sigma\phi$ for every $w\in W$. ∎

It is important to notice that state propositions, as defined in 2.6, are both propositions and co-propositions. It is easy to prove that they have exactly the same semantics under 2.4 and 3.2. The same holds for the co-propositions $[\langle\{g\}\rangle]\varphi$

and the propositions $<\{g\}>\varphi$ which were both abbreviated to $<g>\varphi$.

Moreover, notice that the translation and the co-translation of state propositions coincide. The same does not hold, however, for formulae of the form $<g>\varphi$. Their translation $<\sigma_{ac}^{-1}(g)>\underline{\sigma}(\varphi)$ has a disjunctive flavour (allowed non-determinism) whereas their co-translation $[<\sigma_{ac}^{-1}(g)>]\overline{\sigma}(\phi)$ has a conjuntive flavour (required non-determinism). This difference captures, in fact, the whole point of having introduced the co-language (recall the remark after proposition 2.8). For co-properties, the translation $\overline{\sigma}$ induced by a signature morphism σ satisfies:

Proposition 3.3. Given a signature morphism $\sigma:\Sigma\rightarrow\Sigma'$, for every $\psi\in$ *co-PROP*(Σ), $\Psi\subseteq$ *co-PROP*(Σ), if $\Psi\models_{\Sigma}\psi$ then $\overline{\sigma}(\Psi),\overline{\sigma}(<\perp>\mathbf{true})\models_{\Sigma'}\overline{\sigma}(\psi)$. ∎

Finally, it is possible to define the set of co-properties that a component specified by $<\Sigma,\Phi,\Psi>$ is willing to have in a given context. This set is constituted by the set of co-properties which can be derived from $\overline{\sigma}(\Psi^{\bullet})$ and the state propositions that are ensured, as properties, in that context. Noticing that $\overline{\sigma}(\Psi^{\bullet})\models_{\Sigma'}\psi$ is equivalent to $\overline{\sigma}(\Psi),\overline{\sigma}(<\perp>\mathbf{true})\models_{\Sigma'}\psi$, the definition of a component-of morphism is immediate:

Definition/Proposition 3.4. Given specifications $S=<\Sigma,\Phi,\Psi>$ and $S'=<\Sigma',\Phi',\Psi'>$, a component-of morphism σ from S to S' is a signature morphism $\sigma:\Sigma\rightarrow\Sigma'$, s.t.

(1) $\Phi'\models_{\Sigma'}\underline{\sigma}(\Phi),\underline{\sigma}(\mathrm{loc}(\Sigma))$

(2) $\Phi'^{\bullet}\cap$ *STAT*$(A'),\overline{\sigma}(\Psi),\overline{\sigma}(<\perp>\mathbf{true})\models_{\Sigma'}\Psi'$,

where A' is the set of attributes of Σ'. Specifications and component-of morphisms constitute a category *c-SPEC*. ∎

It is interesting to notice the duality of conditions 1 and 2 of the definition above, in particular, the locality conditions which are associated to the translation of properties ($\underline{\sigma}(\mathrm{loc}(\Sigma))$) and the co-translation of co-properties ($\overline{\sigma}(<\perp>\mathbf{true})$).

3.2 Interconnecting Specifications

The ability of the proposed framework to support the interconnection of components to make complex systems is, as in Goguen's approach to General Systems Theory, characterised by the finite co-completeness of the category of specifications. The specification of a composite system is given by the colimit of its configuration diagram which is guaranteed to exist by the following proposition.

Proposition 3.5. *c-SPEC* is finitely cocomplete. Its initial object is $<\Sigma_0,\emptyset,$ *co-PROP*$(\Sigma_0)>$, where $\Sigma_0=<\emptyset,\{\perp\}>$ is the initial object in *SIGN*. The pushout of $\sigma_1:<\Sigma_0,\Phi_0,\Psi_0>\rightarrow<\Sigma_1,\Phi_1,\Psi_1>$ and $\sigma_2:<\Sigma_0,\Phi_0,\Psi_0>\rightarrow<\Sigma_2,\Phi_2,\Psi_2>$ is given by $\mu_1:<\Sigma_1,\Phi_1,\Psi_1>\rightarrow<\Sigma,\Phi,\Psi>$ and $\mu_2:<\Sigma_2,\Phi_2,\Psi_2>\rightarrow<\Sigma,\Phi,\Psi>$ where:

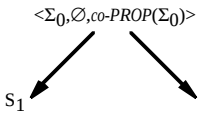
- μ_1 and μ_2 result from the pushout of σ_1 and σ_2 in *SIGN*;
- $\Phi=\underline{\mu}_1(\Phi_1)\cup\underline{\mu}_2(\Phi_2)\cup\underline{\mu}_1(\mathrm{loc}(\Sigma_1))\cup\underline{\mu}_2(\mathrm{loc}(\Sigma_2))$;
- $\Psi=\{\psi\in$ *co-PROP*(Σ): $\Phi^{\bullet}\cap$ *STAT*$(A),\overline{\mu}_1(\Psi_1),\overline{\mu}_1(<\perp>\mathbf{true})\models_{\Sigma}\psi$ and $\Phi^{\bullet}\cap$ *STAT*$(A),\overline{\mu}_2(\Psi_2),\overline{\mu}_2(<\perp>\mathbf{true})\models_{\Sigma}\psi\}$

  where A is the set of attributes of Σ. ∎

Informally, pushouts work as follows:

- Actions are synchronised according to the synchronisation points established by the channel and the morphisms. The synchronisation points defined by such an interconnection are the pairs $g_1 \mid g_2$ which are mapped to the same element of Γ_0, which are given in categorical terms by the limit of the underlying $SET_\perp$ diagram. It is important to notice that independent behaviour of different components is non-stricted interleaved because actions can occur concurrently.
- The properties of the resulting system are given by the union of the translation of the properties and locality axioms of the components.
- The co-properties of the resulting system are the co-propositions that can be proved, separately, from the co-properties of both systems assuming the invariants (over the attributes) of the resulting system. Notice that since we take the translation of the closure of the co-properties of the components, the readiness for the actions that involve only one component consists of the local readiness of that component.

It is important to notice that axioms in Φ_0 and Ψ_0 add no relevant information to the interconnection. In fact, as in [11], it is possible to prove that any interconnection between two specifications can be made through a diagram of the form



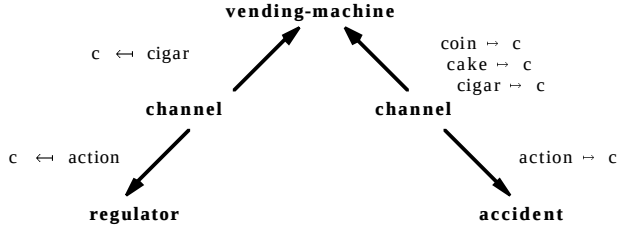
We designate the specifications of the form $<\Sigma,\varnothing,co\text{-}PROP(\Sigma)>$ by channels.

To illustrate composition, let us consider the following systems:

specification regulator **is**
attributes
actions action
axioms [action]**false**
co-axioms

specification accident **is**
attributes outoforder
actions action, breakdown
axioms **beg** $\supset \neg$outoforder
[breakdown]outoforder
co-axioms $\neg$outoforder $\supset$ <action>**true**

The regulator, which has only one action that is never enabled, may be used to prevent an action of another system to occur. For instance, we can interconnect this system with the vending machine presented before in order to obtain a machine with the cigar option blocked. On the other hand, since every non-virtual machine may breakdown at any time, we will simultaneously compose our machine with a component which models this aspect of machine behaviour.



The architecture of the required system is given by the interconnection above, where the signature of channel is $<\varnothing,\{c,\perp\}>$. The resulting specification is given, up to isomorphism, by the specification above. Notice that the readiness to accept the choice of cake is now restricted to the situations in which the machine

is not out of order whereas the readiness to accept the choice of cigar is no longer true. On the other hand, notice that the last two axioms correspond to the translation of the locality axioms of the system components.

specification regulated vending machine **is**
attributes ready, outoforder
actions coin, cake, cigar, breakdown
axioms **beg** $\supset \neg$ready $\wedge \neg$outoforder
ready $\supset$ [coin]**false**
$\neg$ready $\supset$ [cake]**false**
[cigar]**false**
[coin]ready
[cake]$\neg$ready
[breakdown]outoforder
ready $\supset$ [breakdown]ready
$\neg$ready $\supset$ [breakdown]$\neg$ready
co-axioms (ready $\wedge \neg$outoforder) $\supset$ <cake>**true**

3.3 Refinement and Compositionality

Considering that specifications describe the requirements which further developments must fulfill, the refinement of a specification must preserve its properties.

Definition/Proposition 3.6. Given specifications $S=<\Sigma,\Phi,\Psi>$ and $S'=<\Sigma',\Phi',\Psi'>$, a refinement morphism σ from S to S' is a signature morphism $\sigma:\Sigma\rightarrow\Sigma'$, s.t.

(1) $\Phi' \models_{\Sigma'} \underline{\sigma}(\Phi), \underline{\sigma}(\mathrm{loc}(\Sigma))$,

(2) $\Phi'^{\bullet} \cap STAT(A'), \Psi' \models_{\Sigma'} \underline{\sigma}(\Psi)$,

where A' is the set of attributes of Σ'. Specifications and refinement morphisms constitute a category *r-SPEC*. ∎

Refinement allows one to reduce underspecification, since it allows one to take decisions which were before left open and, thus, solved by internal choice.

Taking the semantics of a program P to be the specification $[\![P]\!]$ that consists of the properties it ensures and the co-properties it reflects, refinement morphisms allows us to formalise (and generalise) the usual notion of satisfaction relation between programs and specifications. More concretely, P is a realisation of S via the signature morphism σ iff $\sigma:S\rightarrow[\![P]\!]$ is a refinement morphism. The signature morphism can be seen to record the design decisions that lead from S to P.

In order to illustrate refinement, consider now the following vending machine:

specification vending machine2 **is**
data $<\Sigma_{\mathrm{bool\&nat}}, \Phi_{\mathrm{bool\&nat}}>$
attributes ready:bool, ncoins:nat
actions coin, cake, cigar, collect
axioms **beg** $\supset$ ($\neg$ready $\wedge$ ncoins=0)
ready $\supset$ [coin]**false**
$\neg$ready $\supset$ [cake,cigar]**false**
ncoins=n $\supset$ [coin](ready $\wedge$ ncoins=n+1)
ncoins=n $\supset$ [cake,cigar]($\neg$ready $\wedge$ ncoins=n)
ready $\supset$ [collect](ready $\wedge$ ncoins=0)
$\neg$ready $\supset$ [collect]($\neg$ready $\wedge$ ncoins=0)
co-axioms ($\neg$ready $\wedge$ ncoins<lim) $\supset$ <coin>**true**
ready $\supset$ <cake>**true**
ready $\supset$ <cigar>**true**

Notice that, in this example, we took the extension of the formalism which results from the use of the first order extension of *MAL* and the inclusion of data types in a system specification, more precisely, the inclusion of an algebraic specification of the data types used in each specification.

This specification refines the *vending machine* presented previously. It includes a new attribute (*ncoins*) which represents the actual number of coins inside the deposit of the machine, a new action (*collect*) representing the collection of coins and a readiness axiom for *coin* stating that *coin* is enabled whenever the machine is not ready and the number of coins (*ncoins*) is lower than a given limit (the deposit capacity). In this way, internal nondeterminism over the action *coin* is reduced to the states in which the machine is not ready and the number of coins is greater than the given limit. In these cases, the coin can either be accepted or refused.

It is worthwhile pointing out that the *regulated vending machine* is not a refinement of the initial *vending machine*, because it does not offer the same choices to its environment – more specifically, it does not offer the cigar option.

Refinement morphisms support the process of stepwise refinement of systems with the addition of detail (vertical composition). Given that system development also takes place at the horizontal level through the process of putting together a system from components, it is important to establish properties that relate these two processes (vertical and horizontal).

On the one hand, it is important to confirm that the notion of refinement is "congruent" with the notion of isomorphism in *c-SPEC*. That is, indistinguishable systems (as components of other systems) refine, and are refined exactly by, the same systems. Formally, this corresponds to the fact that isomorphisms in *c-SPEC* are also isomorphisms in *r-SPEC*. Actually, we have a stronger result:

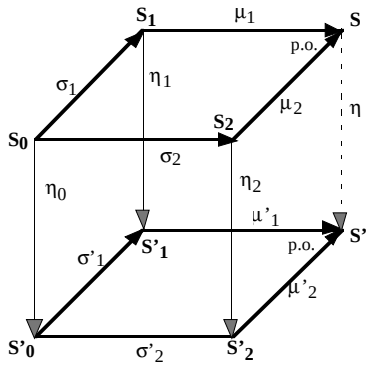
Proposition 3.7. Let $\sigma:\Sigma\rightarrow\Sigma'$ be a signature morphism. $\sigma:S\rightarrow S'$ is an isomorphism in *c-SPEC* iff is an isomorphism in *r-SPEC*. ∎

On the other hand, it is important to investigate compositionality, i.e. the property according to which the refinement of a complex system can be obtained by putting together refinements of its components, what in [12] is called commutativity of horizontal with vertical composition. This is an essential property for supporting incremental development and the reuse of existing components.

Theorem 3.8. Consider the *c-SPEC* diagrams given by $\sigma_1:S_0\rightarrow S_1$, $\sigma_2:S_0\rightarrow S_2$, interconnecting specifications S_1 and S_2 via S_0 and $\sigma'_1:S'_0\rightarrow S'_1$, $\sigma'_2:S'_0\rightarrow S'_2$, interconnecting specifications S'_1 and S'_2 via S'_0. Given refinement morphisms $\eta_0:S_0\rightarrow S'_0$, $\eta_1:S_1\rightarrow S_1'$ and $\eta_2:S_2\rightarrow S_2'$ s.t. $\eta_{0_{ac}}$ is injective and $\eta_0;\sigma'_i=\sigma_i;\eta_i$, for i=1,2, there exists a unique refinement morphism $\eta:S\rightarrow S'$ s.t. $\mu_i;\eta=\eta_i;\mu'_i$, for i=1,2, where $\mu_1:S_1\rightarrow S$ and $\mu_2:S_2\rightarrow S$ result from the pushout of σ_1 and σ_2 and $\mu'_1:S'_1\rightarrow S$ and $\mu'_2:S'_2\rightarrow S$ result from the pushout of σ'_1 and σ'_2. ∎

Notice that, to achieve compositionality it is necessary to ensure that, for every synchronisation point $g_1 \mid g_2$ defined in the upper level, if g'_1 is a possible refinement of g_1 and g'_2 is a possible refinement of g_2 (i.e., $\eta_i(g'_i)=g_i$), then the interconnection specified in the lower level defines the synchronisation point

$g'_1 \mid g'_2$. The following lemma asserts that this is ensured when η_0 is injective.



Lemma 3.9. Given two diagrams $<\sigma_1:\Gamma_1\rightarrow\Gamma_0,\sigma_2:\Gamma_2\rightarrow\Gamma_0>$ and $<\sigma'_1:\Gamma'_1\rightarrow\Gamma'_0, \sigma'_2:\Gamma'_2\rightarrow\Gamma'_0>$ in $SET_\perp$, and morphisms $\eta_0:\Gamma'_0\rightarrow\Gamma_0$, $\eta_1:\Gamma'_1\rightarrow\Gamma_1$ and $\eta_2:\Gamma'_2\rightarrow\Gamma_2$ also in $SET_\perp$ s.t. $\sigma'_i;\eta_0=\eta_i;\sigma_i$, for i=1,2; if η_0 is injective, then for every $g\in\Gamma$, $g'_1\in\Gamma'_1$ and $g'_2\in\Gamma'_2$, if $\eta_1(g'_1)=\mu_1(g)$ and $\eta_2(g'_2)=\mu_2(g)$ then there exists $g'\in\Gamma'$ s.t. $\mu'_1(g')=g_1$ and $\mu'_2(g')=g_2$ and $\eta(g')=g$, where $\mu_1:\Gamma\rightarrow\Gamma_1$ and $\mu_2:\Gamma\rightarrow\Gamma_2$ result from the pullback of σ_1 and σ_2, $\mu'_1:\Gamma'\rightarrow\Gamma'_1$ and $\mu'_2:\Gamma'\rightarrow\Gamma'_2$ result from the pullback of σ'_1 and σ'_2 and $\eta:\Gamma'\rightarrow\Gamma$ results from the universal property of pullbacks. ∎

Proposition 3.10. If C is a channel, $\sigma:C\rightarrow S$ is a component-of morphism and $\eta:S\rightarrow S'$ is a refinement morphism then $\sigma;\eta:C\rightarrow S'$ is a component-of morphism. ∎

From this proposition it follows the following corollary of the theorem 3.8:

Corollary 3.11. Consider the *c-SPEC* diagram given by $\sigma_1:C\rightarrow S_1$, $\sigma_2:C\rightarrow S_2$, interconnecting specifications S_1 and S_2 via the channel C. Given refinement morphisms $\eta_1:S_1\rightarrow S_1'$ and $\eta_2:S_2\rightarrow S_2'$, there exists a unique refinement morphism $\eta:S\rightarrow S'$ s.t. $\mu_i;\eta=\eta_i;\mu'_i$, for i=1,2, where $\mu_1:S_1\rightarrow S$ and $\mu_2:S_2\rightarrow S$ result from the pushout of $\sigma_1;\eta_1$ and $\sigma_2;\eta_2$ and $\mu'_1:S'_1\rightarrow S$ and $\mu'_2:S'_2\rightarrow S$ result from the pushout of σ'_1 and σ'_2. ∎

The corollary states that, given a composite system put together from components through channels, we obtain a system which refines it by using the same channels to interconnect arbitrary refinements of its components.

4 Concluding Remarks

In this paper, we have extended the traditional notion of specification as a theory presentation to include not only the set of sentences that express the properties that are guaranteed of the behaviour of the system but also the sentences (co-properties) that capture the willingness of the system to cooperate with its environment. The distinction between these two sets of sentences was also based on the fact, also observed in [19] in the context of local logics of labelled transition systems, that, whereas certain properties of systems are *preserved* across morphisms, other properties can only be *reflected* in the sense

that they will only be ensured as long as the environment of the component cooperates. Morphisms were defined for the new notion of specification which behave in this way – properties are preserved whereas co-properties are reflected. The resulting category was proved to be finitely co-complete, thus supporting the operation of putting together a system from interconnected components. Another category of specifications was defined for which both properties and co-properties are preserved across morphisms. The morphisms of this category capture refinement relations between specifications, thus supporting vertical structuring. Compositionality (commutativity of horizontal composition wrt vertical composition in the sense of [12]) was also proved meaning that a refinement of a system is obtained by putting together refinements of its components.

The nature of properties and co-properties, i.e. of what is preserved and what is reflected, is determined by the computation and interconnection models that are adopted. In the case that we used for illustration, attributes are local (private). Hence, properties can include invariants proved locally about the attributes of a component, as well as initialisation conditions, effects of actions and restrictions to the occurrence of actions. On the other hand, actions are global (shared) which implies that readiness conditions (the availability of actions) are co-properties: they are directly related to the co-operation with the environment in the sense that an action is only ready to occur when all the components that synchronise in that action make it available. In the context of other interconnection mechanisms, we may have other classes of properties and co-properties. For instance, if local actions are also supported, than readiness for such actions will be treated as properties. If global (shared) attributes are allowed, invariants for those attributes should be treated as co-properties. We are currently working on a model-theoretic characterisation of properties and co-properties which can provide a systematic account of preservation and reflection that can then be incorporated in a generalisation of the notion of institution. This generalisation should then allow us to identify the nature of properties and co-properties in other logics, e.g. temporal logic.

One of the key points of the proposed extension of the notion of specification is the fact that it now reflects more directly the interference between the component and its environment. We are also investigating more closely the relationship between this approach and other styles of specification which, like assumption/commitment or rely/guarantee [1, 5], control the interplay between component and environment.

Finally, and although the paper focused almost exclusively on logical specifications, compositionality results have been proved for an extension of the parallel program design language COMMUNITY [9] that supports the specification of required non-determinism (readiness).

Acknowledgments

We would like to thank Michel Wermelinger and Tom Maibaum for many fruitful interactions.

References

1. M.Abadi and L.Lamport, "Composing Specifications", in *ACM Transactions on Programming Languages and Systems*, 15(1):73-132, January 1993.
2. R.Alur and T.Henzinger, "Local Liveness for Compositional Modeling of Fair Reactive Systems", in P.Wolper (ed), *CAV'95*, LNCS 939, Springer-Verlag 1995, 166-179.
3. H.Barringer, "The Use of Temporal Logic in the Compositional Specification of Concurrent Systems", in A.Galton (ed) *Temporal Logics and their Applications*, Academic Press 1987.
4. R.Burstall and J.Goguen, "Putting Theories Together to Make Specifications", in *Proc. 5th IJCAI*, 1977, 1045-1058.
5. P.Collette, "Design of Compositional Proof Systems Based on Assumption-Commitment Specifications: Application to UNITY", PhD thesis, Université Catholique de Louvain,1994.
6. J.Fiadeiro, "On the Emergence of Properties in Component-Based Systems", in M.Wirsing and M.Nivat (eds), *AMAST'96*, LNCS 1101, Springer-Verlag 1996, 421-443.
7. J.L.Fiadeiro and J.F.Costa, "Mirror, Mirror in my Hand: a duality between specifications and models of process behaviour", in *Mathematical Structures in Computer Science* 6, 1996, 353-373.
8. J.Fiadeiro and T.Maibaum, "Temporal Theories as Modularisation Units for Concurrent System Specification", *Formal Aspects of Computing*, 4:239-272, 1992.
9. J.Fiadeiro and T.Maibaum, "Categorical Semantics of Parallel Program Design", *Science of Computer Programming*, 28:111-138, 1997.
10. J.Fiadeiro and T.Maibaum, "Generalising Interpretations Between Theories in the Context of π-institutions", in G.Burn, S.Gay and M.Ryan (eds), Theory and Formal Methods 1993, Springer-Verlag, 126-147.
11. J.L.Fiadeiro, A.Lopes and T.Maibaum, "Synthesising Interconnections", in Algorithmic Languages and Calculi, R.Bird and L.Meertens (eds), Chapman Hall, in print.
12. J.Goguen and R.Burstall, "CAT, a system for the structured elaboration of correct programs from structured specifications", *Technical Report CSL-118*, SRI International 1980.
13. J.Goguen and R.Burstall, "Institutions: Abstract Model Theory for Specification and Programming", *Journal of the ACM* 39(1):95-146, 1992.
14. B.Jonsson,"Compositional Specifiaction and Verification of Distributed Systems", in *ACM Transactions on Programming Languages and Systems*, 16(2):259-303, 1994.
15. T.Maibaum, "Rôle of Abstraction in Program Development", in H.-J.Kugler (ed) *Information Processing'86*, North-Holland 1986, 135-142.
16. T.Maibaum, "A Logic for the Formal Requirements Specification of Real-Time Embedded Systems", *Forest Research Report*, Imperial College 1987.
17. B.Meyer, "Applying Design by Contract", *IEEE Computer*, Oct.1992, 40-51.
18. C.Stirling, "Modal and Temporal Logics", *Handbook of Logic in Computer Science*, S.Abramsky, D.Gabbay and T.Maibaum (eds), Vol.2, 477-563, Oxford University Press 1992.
19. G.Winskel, "A Compositional Proof System on a Category of Labelled Transition Systems", *Information and Computation* 87:2-57, 1990.
20. S.Zhou, R.Gerth and R.Kuiper, "Transformations Preserving Properties and Properties preserved by Transformations", in E.Best (ed), CONCUR'93, LNCS 715, Springer-Verlag 1993, 353-367.