Electronic Notes in Theoretical Computer Science 4 (1996)

Foundations of Behavioural Specification in Rewriting Logic

Răzvan Diaconescu [1]

Graduate School of Information Science
Japan Advanced Institute of Science and Technology
Ishikawa, Japan

Abstract

We extend behavioural specification based on hidden sorts to rewriting logic by constructing a hybrid between the two underlying logics. This is achieved by defining a concept of behavioural satisfaction for rewriting logic. Our approach is semantic in that it is based on a general construction on models, called *behaviour image*, which uses final models in an essential way. However we provide syntactic characterisations for the for the behavioural satisfaction relation, thus opening the door for shifting recent advanced proof techniques for behavioural satisfaction to rewriting logic. We also show that the rewriting logic behavioural satisfaction obeys the so-called "satisfaction condition" of the theory of institutions, thus providing support for OBJ style modularisation for this new paradigm.

1 Introduction

This research aims at integrating two different semantic approaches on objects and concurrency by internalising behavioural specification [12] to *[conditional] rewriting logic* (abbreviatted **RWL**) [18]. We provide with a logic underlying this integration of the two specification paradigms, which we call **hidden sorted rewriting logic** (abreviatted **HSRWL**), and which is a hybrid between RWL and Goguen's *hidden sorted algebra* (abreviatted **HSA**), i.e., the logic underlying behavioural specification in equational logic. Both HSA and RWL have their origin within the rich tradition of algebraic specification, however their approach to objects and concurrency is quite different. HSRWL aims at providing a unitary framework encoding all important aspects of concurrent objects by combining the HSA approach to behavioural specification of objects (featuring local states, attributes, methods, classes, inheritance, etc.) with the RWL approach to concurrent distributed systems. Thus HSRWL unifies the two constituent sub-logics within a single but flexible framework.

[1] On leave from the Institute of Mathematics of the Romanian Academy.

This is hardly a new idea, the semantics of many multi-paradigm systems where treated similarly, i.e., by unifying the logics involved.

HSRWL is based on a definition of behavioural satisfaction for RWL. We approach behavioural satisfaction for RWL from a semantic angle (as opposed to the syntactic one of [12,14]) by following the general methodology introduced in [3] but adapted to the rather complex case of RWL. This approach highlights a construction on models, called *behaviour image*, using final models; this significantly generalises the concept of "behavioural equivalence" central to the HSA theory. Behaviour images do more than equating [elements but also transitions in the RWL case] under behavioural equivalence, they also add new "behaviour transitions". So we argue that the concept of behaviour image is more fundamental than behaviour equivalence. Another advantage of this semantic approach, also very transparent in [3], is that it is actually independent of sentences, so it can be directly used for many kinds of sentences without regard of their complexity.

As mentioned above, final models play a crucial rôle in our definition of behavioural satisfaction for RWL. We show that final models can be obtained naturally as a (proper) Kan extension, which provides a compact formula for the final models.

We also study a more refined development of behavioural satisfaction for RWL which takes into account the possible transitions between the observations on the states of the system (which are naturally induced by the trasitions at the level of data). This amounts in principle to a *2-version* (we borrow the terminology from 2-categories which play an important technical rôle here) of our theory, and the development of the concepts and results for both the simple and the 2-version is done in parallel. At the end we analyse the relationship between behavioural satisfaction and 2-behavioural satisfaction and provide some sufficient conditions for them to coincide. For example, they coincide whenever there is at most one transition between the elements of data, which in most aplications is a desirable condition.

Our work concentrates at providing the semantic foundations for behavioural specification in RWL, which can be used as a solid logical basis for proving behavioural properties of concurrent distributed systems. This requires a proof theory for the behavioural satisfaction which is actually essential in applications. We give syntactic characterisations for behavioural satisfaction analogous to the original HSA definition of behavioural satisfaction using contexts. This opens the door for using the advanced techniques developed for HSA, such as the so-called "coinduction" of [14].

Behavioural specification supports the powerful module system á la OBJ [15] featuring parameterised programming and module expressions. This is based on the HSRWL institution, thus enabling the direct application of the semantics of module systems developed using institutions [11,8]. The HSRWL institution depends on some conditions on the signature morphisms which were first discovered by Goguen in [10], the most important naturally corresponding to the encapsulation of classes and sub-classes from object-oriented programming. HSRWL adds a new condition corresponding to "encapsulation of structural axioms", this is due to computation modulo structural axioms being treated as a first-class citizen in the definition of RWL (see [18]).

Finally, we hope HSRWL can be used as a framework for the semantics of systems effectively combining behavioural specification and RWL. Such an example is CafeOBJ [9] (whose semantics provide sin fact the main motivation for this work), and in the Appendix we provide a small example written in CafeOBJ whose main purpose is to illustrate in detail the concepts introduced and used in this work.

Acknowledgement

I wish to thank Dr. Dorel Lucanu for carefully reading a preliminary version of this paper and correcting several errors.

Special thanks go to Professor Goguen who encouraged this work, and to Professor Futatsugi and the "CafeOBJ club" at JAIST who provided an excelent environment and useful feedback for this work.

2 Preliminaries

This work assumes certain familiarity from the reader's side with the basics of category theory, also some knowledge and experience with HSA and RWL might prove very useful.

Categorical notations and terminology generally follows [16], with the notably difference that we write composition diagramatically. Also, we may often use the subscript notation rather than the usual bracket notation when evaluating a function at an argument. When using 2-categories we stick with the standard terminology of 0-cells standing for (primitive) objects, 1-cells for arrows between objects, and 2-cells for arrows between arrows. We denote 1-cells by $\rightarrow$ and 2-cells by $\Rightarrow$; this also applies to the standard example of $\mathbb{C}at$, where 0-cells are categories, 1-cells are functors and 2-cells are natural transformations. The class of objects of a category $\mathbb{C}$ is, as usually, denoted as $|\mathbb{C}|$.

Since this work has its origins within the tradition of algebraic specification, we inevitably use a lot of algebraic specification notations and concepts, such as (many sorted) signature, algebras, homorphisms (our terminology and notations being consistent to [12]), but also the poweful concept of institution [11] which constitute the modern level of algebraic specification. We denote an institution $\Im$ by $(\mathbb{S}ign, \text{Mod}, Sen, \models)$, where $\mathbb{S}ign$ is the category of signatures, $\text{Mod}\colon \mathbb{S}ign \rightarrow \mathbb{C}at^{op}$ the model functor, $Sen\colon \mathbb{S}ign \rightarrow \mathbb{C}at$ the sentence functor, and $\models$ is the satisfaction relation. Given a signature morpshim ϕ the reduct functor $\text{Mod}(\phi)$ is often denoted as $\_\!\upharpoonright_{\phi}$ and we often overload ϕ as a short hand notation for $Sen(\phi)$.

The rest of the preliminary section is devoted to the brief presentation of HSA and RWL.

2.1 Behavioural Specification

Hidden sorted algebra was introduced in [12] as an algebraic semantics for the object paradigm capturing its main features such as local states, attributes, methods, classes, inheritance, concurrent distributed operation, etc. The central notion of the HSA algebra approach is that of *behavioural satisfaction*, that is we are interested in the behaviour of the objects rather than in the details of how they are implemented. HSA distinguishes between data, modelled with "visible" sorts, and states, modelled with "hidden" sorts. The recent paper [14] outlines a programme of research on HSA, also giving an overview of some results in this area. By **behavioural specification** we mean specification using HSA.

2.1.1 Hidden sorted algebra

In this section we provide the basic definition of HSA. For reasons of simplicity of presentation HSA assumes a fixed collection of data types given by a many sorted signature (V, Ψ) and a fixed (V, Ψ)-algebra. (V, Ψ, D) is called the **universe of data**. A **hidden sorted signature (over (V, Ψ, D))** is a pair (H, Σ), where H is a set of **hidden** sorts, disjoint from V, and Σ is a $(H \cup V)$ signature, such that

(S1) if $\sigma \in \Sigma_{w,v}$ with $w \in V^*$ and $v \in V$, then $\sigma \in \Psi_{w,v}$; and

(S2) if $\sigma \in \Sigma_{w,v}$ then at most one element of w lies in H.

A **hidden sorted signature morphism** $\phi\colon (H, \Sigma) \to (H', \Sigma')$, is an ordinary signature morphism $\phi\colon \Sigma \to \Sigma'$ such that:

(M1) $\phi(v) = v$ for all $v \in V$ and $\phi(\sigma) = \sigma$ for all $\sigma \in \Psi$,

(M2) $\phi(H) \subseteq H'$, and

(M3) if $\sigma' \in \Sigma'_{w',s'}$ and some sort in w' lies in $\phi(H)$, then $\sigma' = \phi(\sigma)$ for some $\sigma \in \Sigma$.

A **hidden sorted model** over (H, Σ) is a Σ-algebra M such that $M\restriction_{\Psi} = D$ and a **homomorphism of hidden sorted models** $h\colon M \to M'$ is an ordinary homomorphism algebras such that $h\restriction_{\Psi} = 1_D$.

The reader might easily notice that this setup for HSA strongly resembles the setup for *constraint logic* of [6] in that (V, Ψ) can be regarded as a signature of "built-ins", D as a model of "built-ins", and Σ as an extension of (V, Ψ) with "logical" symbols. In this way (V, Ψ, D, Σ) appears as a constraint logic signature, thus enabling the direct use of the model and sentence functors of the constraint logic institution (see [6]) for dealing with variable rather than fixed data types.

2.1.2 Hiding and behaviour in institutions

Although behavioural specification is traditionally developed and studied within the context of equational logic (i.e., algebra), the essential core of the behavioural specification paradigm can be developed in a modern generic style [3], often referred as "institution-independent". In [3] we develop a generic method for defining a behavioural satisfaction relation on top of any satisfaction relation in an institution; in particular, in this paper, we apply this methodolgy to the (mathematically complex) case of RWL.

The main idea of this approach is to provide a *semantic* definition of behavioural satisfaction as opposed to the syntactic traditional one. In the particular case of HSA, the two definitions coincide, but the former one has the advantage that is conceptually more general since it uses an operation on models rather than on sentences. Models usually have simple definitions while sentences sometimes might have complex definitions, moreover many institutions share the same notion of model, but they differ in the sentence part.

In order to describe the notion of behaviourally indistinguishable concisely [3] introduces *behaviour algebras* which are a simple kind of hidden sorted algebras but have a concise formulation of "behaviour". This is somewhat reminiscent of the Lawvere notion of algebra for an algebraic theory [17] in the sense that it is just a functor interpreting a *behaviour signature*, which is a conversion of a hidden sorted signature to a special category. [2] Hidden sorted algebras convert to behaviour algebras and back again and that there are final behaviour algebras. The morphism to the final algebra yields the required behavioural quotient. In [3] we use this to develop a way of taking an institution equipped with some extra data and producing an object version of the institution (or a "hidden" version). The extra data shows how the models of the institution can be viewed as representing behaviour models. Here is the idea in outline. We take an institution $\Im$ plus the extra data and produce an institution $\mathcal{H}(\Im)$, thus

- We have a notion of $\mathcal{H}$ signature, and each $\mathcal{H}$ signature should have a corresponding $\Im$ signature;
- The models of the $\mathcal{H}$ signature are a subcategory of the models of the $\Im$ signature;
- For each model M of an $\mathcal{H}$ signature Σ we can derive a behaviour model;
- We compute an image of this behaviour model by taking the image factorisation of the unique morphism to the final behaviour model (in the particular case of HSA this corresponds to getting the quotient model by equating the elements which have the same observable behaviour);
- We convert this image model back into an $\mathcal{H}$ model $\overline{M}$.
- The satisfaction relation in $\mathcal{H}$ between the model M and any sentence e is defined as the satisfaction in $\Im$ between $\overline{M}$ and e.

All this should work smoothly when one changes signatures, so that we get an institution $\mathcal{H}$ equipped with an institution morphism to $\Im$. This is formalised by the main result of [3], and we use this methodology for defining the behavioural satisfaction in HSRWL.

[2] We will give extended and precise definitions of all these in Section 3.

2.2 Rewriting Logic

Rewriting logic was introduced by Meseguer [18] as a new unifying semantic theory for concurrency. In this paragraph we briefly review the basic concepts of RWL, but the reader is strongly advised to consult [18] for more details.

The syntax of rewriting logic is given by the **rewrite signatures**, which are algebraic theories (Σ, E), where Σ is an algebraic signature [3] and E is a collection of Σ-equations (called *structural equations*). The **sentences** of rewriting logic are conditional rewrite rules modulo E, i.e.,

$$(\forall X)\ [t] \to [t'] \ \textbf{if}\ \ [u_1] \to [v_1] \ldots [u_k] \to [v_k]$$

where t, t', u_i, v_i are Σ-terms with variables X and modulo the equations in E. The left-hand side of **if** is the head of the rule and the right-hand side is the condition of the rule.

The **proof theory** of rewriting logic consists of 4 deduction rules strongly resembling the equational logic deduction rules, the crucial difference being the absence of the *symmetry* rule. This goes to the heart of the philosophy of RWL which is a logic about *changes* rather than equality.

Semantics of RWL is given by its models, called **(rewrite) systems** which provide with an elegant categorical formulation for the concept of concurrent distributed system. In [18] Meseguer presents an impressive list of examples from various areas of concurrency research. A model for the rewrite signature (Σ, E) is given by the interpretation of the corresponding algebraic theory into $\mathbb{C}at$. More concretely, a model A interprets each sort s as a category A_s, and each operation $\sigma \in \Sigma_{w,s}$ as a functor $\sigma_A \colon A_w \to A_s$, where A_w stands for $A_{s_1} \times \ldots \times A_{s_n}$ for $w = s_1 \ldots s_n$. Each Σ-term $t \colon w \to s$ gets a functor $t_A \colon A_w \to A_s$ by evaluating it for each assignment of the variables occuring in t with arrows from the corresponding carriers of A. The satisfaction of an equation $t = t'$ by A is given by $t_A = t'_A$; [4] in particular all structural equations should be satsified by A. A model morphism is a family of functors indexed by the sorts commuting the interpretations of the operations in Σ.

This algebra "enriched" over $\mathbb{C}at$ is a special case of *category-based equational logic* (see [4,5,13]) when letting the category $\mathbb{A}$ of models to be the interpretations of Σ into $\mathbb{C}at$ as abovely described, the category $\mathbb{X}$ of domains to be the category of many sorted sets, and the forgetful functor $\mathcal{U} \colon \mathbb{A} \to \mathbb{X}$ forgetting the interpretations of the operations and the composition between the arrows, i.e., mapping each category to its set of arrows. This enables the use of the machinery of category-based equational logic as a technical aide to the model theory of RWL.

The satisfaction of a rewrite rule $(\forall X)\ [t] \to [t']$ **if** $[u_1] \to [v_1] \ldots [u_k] \to [v_k]$ by a system A has a rather sophisticated definition using the concept of *subequaliser*. Let w be the string of sorts associated to the collection of variables X. Then

[3] Order sorted signature in the full generality, but for presentation simplicity we restrict ourselves to the many-sorted case.
[4] This definition extends without difficulty to conditional equations.

$$A \models (\forall X)\ [t] \to [t']\ \textbf{if}\ \ [u_1] \to [v_1] \dots [u_k] \to [v_k]$$

iff there exists a natural transformation $J_A; t_A \Rightarrow J_A; t'_A$ where $J_A\colon\ Subeq((u_{iA}, v_{iA})_{i\in\overline{1,k}}) \to A_w$ is the subequaliser functor, i.e., the functor component of the final object in the category having pairs $(Dom(S) \overset{S}{\to} A_w, (S; u_{iA} \overset{\alpha_i}{\Rightarrow} S; v_{iA})_{i\in\overline{1,k}})$ as objects and functors H such that $H; S' = S$ and $H\alpha' = \alpha$ as arrows.

The satisfaction relation between the sentences and the syntax of rewriting logic gives an institution in which the signature morphisms are morphisms of algebraic theories and the translation of the quantifiers and terms (modulo structural axioms) along the signature morphisms is the same as in the institution of equational logic modulo axioms (denoted *ELM* in [7]).

3 (2-)Behaviour Signatures and Systems

Behaviour systems provide a concise formulation of "behaviour" for concurrent systems by converting a hidden sorted rewrite model (system) into another kind of system which is mathematically simpler. Behaviour systems generalise behaviour algebras of [3] in the same way the models of rewriting logic generalise the models of equational logic, i.e., general algebra.

Definition 3.1 *A* **(2-)behaviour signature** *is a (2-)category with distinguished 0-cell d such that there are no 1-cells whose source is d except the identity; we let a* **morphism of (2-)behaviour signatures** *be a (2-)functor preserving the distinguished 0-cell and such that distinguished 0-cells are the only 0-cells mapped to distinguished 0-cells.*

We write a (2-)behaviour signature as a pair (C, d) where C is a (2-)category and d is its distinguished 0-cell. We use h and h' for 0-cells different from d, and we use e for 1-cells.

Notice that 2-behaviour signatures are simple behaviour signatures by ignoring the 2-cells. Similarly for signature morphisms.

Fact 3.2 *Each hidden sorted signature (H, Σ) over (V, Ψ, D) gives rise to a 2-behaviour signature (C, d) where*

- *$|C| = H \cup \{d\}$; and*
- *C is freely generated by the family of sets of 2-cells $C[h, n] = \{\langle\sigma, a\rangle \overset{r}{\Rightarrow} \langle\sigma, a'\rangle \mid \sigma \in \Sigma_{vhv',n}, v, v' \in V^*, r \in D_{vv'}(a, a')\}$ where $h \in H$ and $n \in H \cup \{d\}$.*

Each hidden sorted signature morphism $\phi\colon (H, \Sigma) \to (H', \Sigma')$ gives rise to a 2-behaviour signature morphism ϕ by letting $\phi(\langle\sigma, a\rangle \overset{r}{\Rightarrow} \langle\sigma, a'\rangle) = \langle\phi(\sigma), a\rangle \overset{r}{\Rightarrow} \langle\phi(\sigma), a'\rangle$.

In the following we provide (2)-functorial semantics for behaviour systems in the style of Lawvere functorial semantics for algebraic theories [17].

Definition 3.3 *Let (C, d) be a (2-)behaviour signature and D be a category. A* **(2-)behaviour system** *over (C, d, D) is a (2-)functor $A\colon\ C \to \mathbb{C}at$ such that $A_d = D$. A morphism of (2-)behaviour systems is a (2-)natural transformation* *such that its component at d is the identity on D. By $\mathbb{B}Sys(C, d, D)$ we denote the category of behaviour systems and their morphisms and by $2\mathbb{B}Sys(C, d, D)$ the category of 2-behaviour systems and 2-behaviour morphisms.*

3.1 The Final Behaviour System

As emphasised in [3,12], final models play a crucial role in the semantics of behavioural specification. Following [3], in Section 4.2 we use final models for providing a semantic definition for behavioural satisfaction in RWL. We concentrate here on the final behaviour system (in both the simple and the more complex version) which we obtain as a Kan-extension.

Theorem 3.4 *For any (2-)behaviour signature (C, d, D) there exists the final (terminal) (2-)behaviour system $\boldsymbol{B}$ which can be obtained as the right (2-)Kan extension of d along D.*

Proof. If d and D are regarded as (2-)functors from the final category $\star$ consisting of one identity cell, then the final (2-)behaviour system over (C, d, D) is just the right (2-)Kan extension of d along D.

Let's first discuss the simpler case of ordinary behaviour systems. The right Kan-extension exists since $\mathbb{C}at$ is small complete, i.e., it has all small limits (Corrolary 2 pg. 235 of [16]).

$$\begin{tikzcd}
\star \arrow[r, "d"] \arrow[dr, "D"'] & C \arrow[d, "\boldsymbol{B}"] \\
 & \mathbb{C}at
\end{tikzcd}$$

A slightly subtle point concerns the equality $d; \boldsymbol{B} = D$ for $\boldsymbol{B}$ the right-Kan extension of d along D. This is equivalent with the condition that $\boldsymbol{B}$ is really a behaviour system, i.e., $\boldsymbol{B}(d) = D$. This follows directly by Corollary 3 pg. 235 of [16] because d is full (since there are no arrows in C out of d except the identity) and faithful as a functor.

The 2-case can be treated similarly by using (the dual of) Theorem 6.7.7 from [2] [5] applied to the $\mathbb{C}at$-enriched case, for example.

For understanding the "behavioural" structure of $\boldsymbol{B}$, we need to explicitate its construction (see Corollary 2 pg. 235 of [16] for the case of ordinary categories):

Corollary 3.5 *The structure of the final 2-behaviour system $\boldsymbol{B}$ is given by:*

(i) *for each 0-cell $n \in |C|$, $\boldsymbol{B}_n$ is the functor category $D^{C(n,d)}$,*

(ii) *for each 1-cell $e \in |C(n, n')|$, $\boldsymbol{B}_e$ is $D^{C(e,d)}$, i.e., the functor $D^{C(n,d)} \to D^{C(n',d)}$ given by $(\boldsymbol{B}_e)(f)_c = f_{e;c}$ where $f \in D^{C(n,d)}$ and $c \in C(n', d)$, and*

(iii) *(**for the 2-case only**) for each 2-cell $e \overset{r}{\Rightarrow} e'$, $\boldsymbol{B}_r$ is $D^{C(r,d)}$, i.e., the natural transformation $D^{C(e,d)} \Rightarrow D^{C(e',d)}$ given by $((\boldsymbol{B}_r)_f)_c = f_{rc}$ for each $f \in D^{C(n,d)}$ and each $c \in C(n', d)$.*

[5] Generalising the standard Kan-extension existence result to enriched category theory.

Notice that the behaviour system underlying the final 2-behaviour system is not final! One can easily notice this when comparing the definition of $\boldsymbol{B}_n$ (for n arbitrary 0-cell) in each case; in the 2-case this is a proper functor category while in the simple case it is just a power category (having of course "more" objects and arrows). This difference finally amounts in principle to different concepts of behavioural satisfaction for RWL.

3.2 The Image Behaviour System

We now turn to the definition of image behaviour which plays the most important rôle for the definition of behavioural satisfaction. In the case of RWL this provides a non-trivial generalisation of HSA behavioural quotients because in RWL the behaviour image system apart of equating elements and transitions under the behavioural equivalence relation, also adds new transitions to the original model. This is formally achieved by using a rather abstract formulation, the notion of *image factorisation system* as defined in [1]. In fact our factorisation system for behaviour systems is inherited from one of the factorisation systems in $\mathbb{C}at$ as follows:

Fact 3.6 *The category of systems over (C, d, D) has a canonical image factorisation system $(\mathcal{E}_C, \mathcal{M}_C)$ with $\mathcal{E}_C = \{e$ morphism of (2-)behaviour systems $\mid$ e_h surjective on objects for any $h\}$ and $\mathcal{M}_C = \{m$ morphism of (2-)behaviour systems $\mid m_h$ full, faithful and injective on objects for any $h\}$.*

Proof. Each component of a morphism of (2-)behaviour systems factors in $\mathbb{C}at$ through the image factorisation system $(\mathcal{E}, \mathcal{M})$ of $\mathbb{C}at$ where $\mathcal{E} = \{e \mid e$ surjective on objects$\}$ and $\mathcal{M} = \{m \mid m$ full, faithful and injective on objects$\}$, then we use the Diagonal Fill-in Property for image factorisation systems to define the image behaviour system on arrows.

More specifically, suppose $m\colon A \to \boldsymbol{B}$ is a morphism of systems and $e\colon n \to n'$ a 1-cell in C, so that $A_e\colon A_n \to A_{n'}$ and $\boldsymbol{B}_e\colon \boldsymbol{B}_n \to \boldsymbol{B}_{n'}$. Then we factorise $m_n\colon A_n \to \boldsymbol{B}_n$ to get an intermediate $\overline{A}_n$, similarly for n'. We then use the fill in property to get $\overline{A}_e\colon \overline{A}_n \to \overline{A}_{n'}$. This gives the factorisation of m with image system $\overline{A}$.

Only for the case of of 2-behaviour systems, we also need to define $\overline{A}_r$ for any 2-cell $e \overset{r}{\Rightarrow} e'$. This is given by simply restricting $\boldsymbol{B}_r$.

Definition 3.7 *Given a behaviour system A, its* **image (2-)behaviour system** *is $\overline{A}$, where*

$$\begin{tikzcd} A \arrow[r] & \overline{A} \arrow[r] & \boldsymbol{B} \end{tikzcd}$$

is the factorisation of the unique behaviour (2-)system morphism $A \to \boldsymbol{B}$, where $\boldsymbol{B}$ is the final (2-)behaviour system.

The final semantics concept of image behaviour system (or model) is dual to the concept of *reachable submodel* from the initial semantics. Reachable submodels can be obtained in the same way by factoring the unique morphism from the initial model to the corresponding model.

Any morphism $\phi\colon (C, d) \to (C', d)$ of behaviour signatures determines a functor

$\_\lceil_\phi \colon \mathbb{B}Sys(C', d, D) \to \mathbb{B}Sys(C, d, D)$ mapping a behaviour system A' to $\phi; A'$ and any morphism f' of behaviour systems to $\phi f'$ (i.e., the vertical composition between ϕ as a functor and f' as a natural transformation).

Corollary 3.8 *For any morphism $\phi \colon (C, d) \to (C', d)$ of (2-)behaviour signatures, $\lceil_\phi$ is a morphism of factorisation systems $(\mathcal{E}_{C'}, \mathcal{M}_{C'}) \to (\mathcal{E}_C, \mathcal{M}_C)$, i.e., $\mathcal{E}_{C'}\lceil_\phi \subseteq \mathcal{E}_C$ and $\mathcal{M}_{C'}\lceil_\phi \subseteq \mathcal{M}_C$.*

Lemma 3.9 *Let $\Phi \colon I \to J$ be a functor surjective on objects and D be any category. Then the functor $D^\Phi \colon D^J \to D^I$ is faithful and injective on objects.*

Corollary 3.10 *Let $\phi \colon (C, d) \to (C', d)$ be a morphism of (2-)behaviour signatures with $\phi(h, d) \colon C(h, d) \to C'(\phi(h), d)$ surjective for any h. Let $\boldsymbol{B}$ be the final (2-)behaviour system over (C, d, D) and $\boldsymbol{B}'$ be the final system over (C', d, D). Then the unique morphism of (2-)behaviour systems $m \colon \boldsymbol{B}'\lceil_\phi \to \boldsymbol{B}$ is injective in all components, i.e., it belongs to $\mathcal{M}_C$.*

Proof. Fix $h \in |C| - \{d\}$. $(\boldsymbol{B}'\lceil_\phi)_h = \boldsymbol{B}'_{\phi(h)} = D^{C'(\phi(h),d)}$. Then by applying the previous lemma for $\phi(h, d)$ in the rôle of Φ, we get that m_h is faithful and injective on objects, therefore $m \in \mathcal{M}_C$.

Fact 3.11 *Any morphism of (2-)behaviour signatures generated by a hidden sorted signature morphism has the surjectivity property of the previous Corollary.*

Proof. By using the condition (M3) of the definition of morphisms of hidden sorted signatures.

4 Hidden Sorted Rewriting Logic

In this section we use the technique previously developed for defining the basic ingredients of HSRWL, such as signatures, models, and satisfaction of sentences by models. We finally show that HSRWL is an institution.

4.1 Signatures and Models

Definition 4.1 *A* **hidden sorted rewrite signature** *is given by (H, Σ, E), where (H, Σ) is a hidden sorted signature over (V, Ψ, D) with the difference that D is a rewrite model for (V, Ψ) rather than an algebra, and E is a collection of Σ-equations. A* **morphism** *$\phi \colon (H, \Sigma, E) \to (H', \Sigma', E')$ of hidden sorted rewrite signatures is a morphism of hidden sorted signatures $(H, \Sigma) \to (H', \Sigma')$ such that the following is satisfied:*

(M4) $\phi(E) \models_{\Sigma'} E'$

A **hidden sorted rewrite model** *M for a hidden sorted rewrite signature (H, Σ, E) over (V, Ψ, D) is just a (Σ, E)-rewrite model such that $M\lceil_\Psi = D$.*

Condition (M4) corresponds to the “encapsulation of structural axioms” and together with (M3) play the crucial rôle in obtaining the Satisfaction Condition for HSRWL.

Fact 4.2 *Given a hidden sorted rewrite model M for a signature (H, Σ, E), one can canonically extract a 2-behaviour system $[M]$ for the 2-behaviour signature (C, d) generated by (H, Σ), by letting*

- $[M]_h = M_h$ *for each* $h \in H$,
- $[M]_{\langle\sigma,a\rangle} = \sigma_M(\_, a)\colon M_h \to M_{h'}$ *for each* $\sigma \in \Sigma_{vhv',h'}$ *and* $a \in |D_{vv'}|$, *and*
- $([M]_r(b) = \sigma_M(r, 1_b)$ *for each* $r \in D_{vv'}(a, a')$ *and* $b \in |M_h|$.

This can be easily extended to a forgetful functor $[\_]$ from the category of hidden sorted rewrite models to the category of 2-behaviour systems.

Lemma 4.3 *Let A be a (H, Σ)-rewrite model and $m\colon [A] \to B$ be a morphism of behaviour systems. There is an unique morphism of (H, Σ)-rewrite models $m^\sharp\colon A \to B^\sharp$ such that $[m^\sharp] = m$.*

Proof. Because $[m^\sharp]$ should be m, for any $h \in H$ we take $B^\sharp_h$ to be B_h and $m^\sharp_h$ to be m_h. If $\sigma \in \Sigma_{vhv',h'}$, then, for all $a \in |D_v|$, $b \in B_h$, $\sigma_{B^\sharp}(b, a)$ is $B_{\langle\sigma,a\rangle}(b)$. If $\sigma \in \Sigma_{v,h}$, $v \in V^*$, $h \in H$, then, for all $a \in D_v$, $\sigma_{B^\sharp}(a)$ is defined as $m_h(\sigma_A(a))$. These define a hidden sorted rewrite model $B^\sharp$ and a morphism $m^\sharp\colon A \to B^\sharp$. Notice that $m^\sharp$ is indeed a morphism of hidden sorted models because of the naturality of m and of the definition of the interpretations of the operations $\sigma \in \Sigma_{v,h}$, $v \in V^*$, $h \in H$, in $B^\sharp$.

4.2 Behavioural Satisfaction for Rewriting Logic

In this section we define the behavioural satisfaction relation for HSRWL by using the behaviour image system rather than the 2-behaviour system. 2-behavioural satisfaction is discussed in a section below.

Definition 4.4 *Let (H, Σ, E) be a hidden sorted signature. Then for each rewrite model M, its* **behaviour image** *$\overline{M}$ is defined as $\overline{[M]}^\sharp$, where*

- *$\overline{[M]}$ is the image of the unique morphism of behaviour systems $[M] \to \boldsymbol{B}$, and*
- *$\overline{[M]}^\sharp$ is obtained by lifting the canonical map $[M] \to \overline{[M]}$ back to (H, Σ)-rewrite models.*

A (H, Σ, E)-rewrite model M **behaviourally satisfies** *a sentence $[\rho]$ iff $\overline{M} \models \rho$. We write this as $M \models\!\!\!\equiv_{(H,\Sigma,E)} [\rho]$ (or just $M \models\!\!\!\equiv [\rho]$ for short).*

Sentences $[\rho]$ in this case can be either (possibly conditional) rules or equations modulo the structural equations E. The notation $[\rho]$ extends the usual notation for equivalence classes of terms modulo E to rules and equations in the obvious way.

Theorem 4.5 [Satisfaction Condition for HSRWL] *Let $\phi\colon (H, \Sigma, E) \to (H', \Sigma', E')$ be a morphism of hidden sorted rewrite signatures, M' be a (H', Σ', E')-rewrite model, and ρ be a Σ-rule or a Σ-equation. Then*

$$M' \models\!\!\!\equiv_{(H',\Sigma',E')} \phi([\rho]) \quad \textit{iff} \quad M'\!\restriction_\phi \models\!\!\!\equiv_{(H,\Sigma,E)} [\rho]$$

Proof. This proof follows the idea of the proof of Theorem 15 of [3]. Since there is no danger of confusion by ϕ we will denote both the hidden sorted signature morphism $(H, \Sigma) \to (H', \Sigma')$ and the corresponding behaviour signature morphism.

We first show the naturality of the behaviour image wrt the signature morphisms, i.e., that

$$\overline{M'}\restriction_\phi = \overline{M'\restriction_\phi}$$

Consider the following diagram of behaviour systems:

$$\begin{tikzcd}
\phi;[M'] \arrow[r, "e^1"] \arrow[d, equal] & \phi;\overline{[M']} \arrow[r, "m^1"] & \phi;\boldsymbol{B}_{\Sigma'} \arrow[d, "m"] \\
{[M'\restriction_\phi]} \arrow[r, "e^2"'] & \overline{[M'\restriction_\phi]} \arrow[r, "m^2"'] & \boldsymbol{B}_\Sigma
\end{tikzcd}$$

We have that $e^1 \in \mathcal{E}_C$ and $m^1 \in \mathcal{M}_C$ by Corollary 3.8, $e^2 \in \mathcal{E}_C$ and $m^2 \in \mathcal{M}_C$ by definition, and $m \in \mathcal{M}_C$ by Corollary 3.10. The uniqueness of the factorisation gives us $\phi;\overline{[M']} = \overline{[M'\restriction_\phi]}$ (modulo a canonical isomorphism). By applying Lemma 4.3 for $e^1 = e^2$, we get that $\overline{M'}\restriction_\phi = \overline{M'\restriction_\phi}$.

Now we can proceed with the proof of the Satisfaction Condition. We may assume that ρ is a rule, the case when ρ is an equation can be treated similarly and it might be more familiar from the HSA.

$$\begin{aligned}
M' \models\!\!\!\equiv_{(H',\Sigma',E')} \phi([\rho]) \quad & \text{iff} \quad M' \models_{\Sigma'} E' \ \text{ and } \ \overline{M'} \models_{\Sigma'} \phi(\rho) \\
& \text{iff} \quad M'\restriction_\phi \models_\Sigma E \ \text{ and } \ \overline{M'} \models_\Sigma \phi(\rho) \\
& \qquad \text{((M4) and the CBEL Satisfaction Condition)} \\
& \text{iff} \quad M'\restriction_\phi \models_\Sigma E \ \text{ and } \ \overline{M'}\restriction_\phi \models_\Sigma \rho \\
& \qquad \text{(RWL Satisfaction Condition)} \\
& \text{iff} \quad M'\restriction_\phi \models_\Sigma E \ \text{ and } \ \overline{M'\restriction_\phi} \models_\Sigma \rho \\
& \qquad \text{(by the previous argument)} \\
& \text{iff} \quad M'\restriction_\phi \models\!\!\!\equiv_{(H,\Sigma,E)} [\rho] \\
& \qquad \text{(by definition).}
\end{aligned}$$

4.3 Towards a Proof Theory for RWL Behavioural Satisfaction

In this section we develop some syntactic characterisations of the behavioural satisfaction relation defined in the previous section. This opens the door for using recent advanced techniques developed for HSA (such as the "coinduction" of [14]) for supporting proofs of properties of behavioural specification in RWL.

Proposition 4.6 *Let (H, Σ, E) be a hidden sorted rewrite signature. Then for each (H, Σ, E)-model M and for any rule $[t] \to [t']$,*

$$M \models\!\equiv [t] \to [t'] \;\; \textit{implies} \;\; M \models c(t) \to c(t') \;\; \textit{for all contexts} \;\; c \;\; \textit{of visible sort}$$

Proof. Given a hidden sort $h \in H$, there is a canonical one-one correspondence

$$\{c(z) \mid c \text{ context of visible sort with the variable } z \text{ of sort } h\} = C(h,d)$$

where (C,d) is the behaviour signature determined by (H,Σ,E).

$M \models\!\equiv [t] \to [t']$ means that $\overline{M} \models t \to t'$, which means there exists a natural transformation $\alpha\colon t_{\overline{M}} \Rightarrow t'_{\overline{M}}$. We need to find a natural transformation $\alpha^c\colon c(t)_M \Rightarrow c(t)_M$ for each visibile context $c \in C(h,d)$. Assume $t,t'\colon w \to h$. Then for each $p \in |M_w|$ we define

$$\alpha^c_p = c_{\overline{M}}(\alpha_{\overline{p}}),$$

where $\overline{p}$ is the image of p in $\overline{M}_w$ via the canonical map $M \to \overline{M}$. We have to show the naturality of α^c_p, i.e., the commutativity of the following diagram:

$$\begin{tikzcd}
p \arrow[d, "\varphi"'] & c_M(t_M(p)) \arrow[r, "\alpha^c_p"] \arrow[d, "c_M(t_M(\varphi))"'] & c_M(t'_M(p)) \arrow[d, "c_M(t'_M(\varphi))"] \\
p' & c_M(t_M(p')) \arrow[r, "\alpha^c_{p'}"'] & c_M(t'_M(p'))
\end{tikzcd}$$

for all $\varphi \in M_w$. But this follows because c has visible sort, which means that $c_M(t_M(\varphi)) = c_{\overline{M}}(t_{\overline{M}}(\overline{\varphi})) \in D$, and by the naturality of α.

Theorem 4.7 *Let (H,Σ,E) be a hidden sorted rewrite signature over (V,Ψ,D) with D partial order. Then for each (H,Σ,E)-model M and for any rule $[t] \to [t']$,*

$$M \models\!\equiv [t] \to [t'] \;\; \textit{iff} \;\; M \models c(t) \to c(t') \;\; \textit{for all contexts} \;\; c \;\; \textit{of visible sort}$$

Proof. Assume $M \models c(t) \to c(t')$. By definition $M \models\!\equiv [t] \to [t']$ is equivalent to $\overline{M} \models t \to t'$. So we have to find a natural transformation $\overline{\alpha}\colon t_{\overline{M}} \Rightarrow t'_{\overline{M}}$.

Assume that $t,t'\colon w \to h$, and as in Proposition 4.6 notice the one-one correspondence between the contexts of argument h and $C(h,d)$. So, by hypothesis we know that for each $c \in C(h,d)$, there exists $\alpha^c\colon c(t)_M \to c(t')_M$.

In the virtue of the definition of $\overline{M}$ we may consider $\overline{M}_h$ as a full subcategory of $\boldsymbol{B}_h$. The for each $p \in |M_h|$ we define

$$\overline{\alpha}_{\overline{p}} = \{\alpha^c_p\}_{c \in C(h,d)}$$

The correctenss of this definition for $\overline{\alpha}\colon t_{\overline{M}} \Rightarrow t'_{\overline{M}}$ is assured by the fact that because $\overline{M}_h \subseteq \boldsymbol{B}_h$ is full, the canonical map $M \to \overline{M}$ is surjective on the elements (i.e., objects) of the carriers and it is a congruence, therefore for each $c \in C(h,d)$, if $\overline{p} = \overline{p_1}$ then α^c_p and $\alpha^c_{p_1}$ have the same source and target in D, so they are equal.

We still have to prove the naturality of $\overline{\alpha}$, i.e., the commutativity of

$$\begin{tikzcd}
\overline{p} \arrow[d, "\overline{\varphi}"'] & t_{\overline{M}}(\overline{p}) \arrow[r, "\overline{\alpha}_{\overline{p}}"] \arrow[d, "t_{\overline{M}}(\overline{\varphi})"'] & t'_{\overline{M}}(\overline{p}) \arrow[d, "t'_{\overline{M}}(\overline{\varphi})"] \\
\overline{p'} & t_{\overline{M}}(\overline{p'})) \arrow[r, "\overline{\alpha}_{\overline{p'}}"'] & t'_{\overline{M}}(\overline{p'})
\end{tikzcd}$$

in $\overline{M}_h$ for all $\overline{\varphi} \in \overline{M}_w(\overline{p}, \overline{p'})$. If we consider this diagram in $\boldsymbol{B}_h$, then it commutes componentwise, i.e.,

$$\begin{tikzcd}
\overline{p} \arrow[d, "\overline{\varphi}"'] & c_M(t_M(p)) \arrow[r, "\alpha^c_p"] \arrow[d, "c_{\overline{M}}(t_{\overline{M}}(\overline{\varphi}))"'] & c_M(t'_M(p)) \arrow[d, "c_{\overline{M}}(t'_{\overline{M}}(\overline{\varphi}))"] \\
\overline{p'} & c_M(t_M(p')) \arrow[r, "\alpha^c_{p'}"'] & c_M(t'_M(p'))
\end{tikzcd}$$

commutes for each $c \in C(h, d)$ because in D there is at most one arrow $c_M(t_M(p)) \to c_M(t'_M(p'))$.

Corollary 4.8 *For any hidden sorted rewrite signature (H, Σ, E) over (V, Ψ, D) with D partial order, any model M, and any rule $[t] \to [t']$, we have that*

$$M \models\!\equiv [t] \to [t'] \quad \textit{if} \quad M \models t \to t'$$

Proof. By the soundness of RWL $M \models [t] \to [t']$ implies that $M \models c(t) \to c(t')$ for all contexts c of visible sort, which further implies $M \models\!\equiv [t] \to [t']$ by Theorem 4.7.

Corollary 4.9 *Let (H, Σ, E) be a hidden sorted rewrite signature over (V, Ψ, D) with D partial order and $[R]$ be a rewrite theory. Then*

$$[R] \models\!\equiv [t] \to [t'] \quad \textit{iff} \quad [R] \vdash [c(t)] \to [c(t')] \quad \textit{for all contexts} \quad c \quad \textit{of visible sort}$$

Proof. By the completeness of RWL, $[R] \vdash [c(t)] \to [c(t')]$ is equivalent to $[R] \models [c(t)] \to [c(t')]$.

First assume $[R] \models\!\equiv [t] \to [t']$ and consider a (H, Σ, E)-model for R. By Corollary 4.8, $M \models\!\equiv [R]$, therefore $M \models\!\equiv [t] \to [t']$, which by Proposition 4.6 implies $M \models c(t) \to c(t')$.

For the converse assume $[R] \models [c(t)] \to [c(t')]$ and consider a model M such that $M \models\!\equiv [R]$. Then $\overline{M} \models R$, which implies $\overline{M} \models c(t) \to c(t')$, which means $M \models\!\equiv [c(t)] \to [c(t')]$. But since c is of visible sort, this is the same with $M \models c(t) \to c(t')$ which by Theorem 4.7 implies $M \models\!\equiv [t] \to [t']$.

5 2-Behavioural Satisfaction

By using 2-behaviour system instead of behaviour systems, we can re-use Definition 4.4 for defining **2-behavioural satisfaction**, which we denote by $\models\!\equiv^2$. Similarly to Theorem 4.5 the ordinary behavioural satisfaction case, we have a Satisfaction Condition for $\models\!\equiv^2$:

Theorem 5.1 *Let $\phi\colon (H,\Sigma,E) \to (H',\Sigma',E')$ be a morphism of hidden sorted rewrite signatures, M' be a (H',Σ',E')-rewrite model, and ρ be a Σ-rule or a Σ-equation. Then*

$$M' \models^2_{(H',\Sigma',E')} \phi([\rho]) \quad \textit{iff} \quad M'\!\restriction_\phi \models^2_{(H,\Sigma,E)} [\rho]$$

The rest of this section is devoted to the study of the relationship betwen behavioural satisfaction and the 2-behavioural satisfaction relation. Fix a signature (H,Σ,E) over (V,Ψ,D), and let $\boldsymbol{B}$ denote the final behaviour system, $\boldsymbol{B}^2$ the final 2-behaviour system.

For any model M, let $\overline{M}$ denote its image behaviour system and $\overline{\overline{M}}$ denote its image 2-behaviour system.

Definition 5.2 *Let $h \in H$. Then $a, a' \in M_h$ are* **behaviourally equivalent** *iff $\overline{a} = \overline{a'}$, i.e., they get identified by the canonical map $M \to \overline{M}$; we denote this by $a \sim a'$.*

Similarly, a and a' are **2-behaviourally equivalent** *iff $\overline{\overline{a}} = \overline{\overline{a'}}$; we denote this by $a \approx a'$.*

Fact 5.3 *Both $\sim$ and $\approx$ are Σ-congruences.*

Lemma 5.4 *Let $h \in H$ and let $a \xrightarrow{\varphi} b, a' \xrightarrow{\varphi} b' \in M_h$. Then,*

(i) *$a \sim a'$ iff $[M]_c(a) = [M]_c(a')$ for all $c \in |C(h,d)|$,*

(ii) *$a \approx a'$ iff $a \sim a'$ and $([M]_r)_a = ([M]_r)_{a'}$ for all $c \overset{r}{\Rightarrow} c' \in C(h,d)$,*

(iii) *$\varphi \sim \varphi'$ iff $[M]_c(\varphi) = [M]_c(\varphi')$ for all $c \in |C(h,d)|$, and*

(iv) *$\varphi \approx \varphi'$ iff $a \approx a', b \approx b'$, and $\varphi \sim \varphi'$.*

Proof. By explicitating the action of the unique (2-) behaviour system morphism from $[M]$ to $\boldsymbol{B}$ (or $\boldsymbol{B}^2$ in the 2-case).

Proposition 5.5 *There exists an unique morphism of hidden sorted rewrite models $\beta\colon \overline{\overline{M}} \to \overline{M}$ such that for each $h \in H$ the following diagram commutes in $\mathbb{C}at$:*

$$\begin{tikzcd}
M_h \arrow[r] \arrow[dr] & \overline{\overline{M}}_h \arrow[r] \arrow[d, "\beta_h"] & \boldsymbol{B}^2_h \arrow[d] \\
& \overline{M}_h \arrow[r] & \boldsymbol{B}_h
\end{tikzcd}$$

where $\boldsymbol{B}^2 \to \boldsymbol{B}$ is the unique behaviour system morphism from $\boldsymbol{B}^2$ (regarded as a behaviour system) to $\boldsymbol{B}$.

Proof. We define $\beta_h(\overline{\overline{a}}) = \beta_h(\overline{a})$ for each $a \in M_h$. This covers the definition of β_h on objects and arrows originating from M_h. Let now $\overline{\overline{a}} \xrightarrow{\varphi} \overline{\overline{a'}}$ such that there is no arrow $a \xrightarrow{\theta} a'$ in M_h with $\overline{\overline{\theta}} = \varphi$. then because $\overline{\overline{M}}_h \subseteq \boldsymbol{B}^2_h$ is full, $\varphi \in \boldsymbol{B}^2_h$. Let φ' be its image in $\boldsymbol{B}_h$ via the canonical map $\boldsymbol{B}^2_h \to \boldsymbol{B}_h$. Since $\overline{M}_h$ is full, $\varphi' \in \overline{M}_h$, therefore we may define $\beta_h(\varphi) = \varphi'$. Routine verifications will show that β_h is functor and that β is a model morphism.

Corollary 5.6 *Using the same notations as above,*

- *β is full (in all components) iff $\boldsymbol{B}^2_h \to \boldsymbol{B}_h$ is full for each $h \in H$, and*

- *β is isomorphism iff $\boldsymbol{B}^2_h \to \boldsymbol{B}_h$ is a full subcategory for each $h \in H$.*

Corollary 5.7 *If $\boldsymbol{B}^2_h \to \boldsymbol{B}_h$ is full for each $h \in H$, then for each hidden sorted rewrite model M and each unconditional equation or rewrite rule $[\rho]$,*

$$M \models\!\equiv [\rho] \quad \textit{if} \quad M \models\!\equiv^2 [\rho]$$

Proof. In the virtue of the definition of behavioural satisfaction (either in the simple case and in the 2-case), we have to prove that $\overline{M} \models \rho$ if $\overline{\overline{M}} \models \rho$.

If $[\rho]$ is an equation, then it is easy to check that by using the classical argument that quotients preserve the validity of unconditional equations. This is due to the fact that $\beta\colon \overline{\overline{M}} \to \overline{M}$ is a surjective on objects and full homomorphism.

If ρ is a rewrite rule $t \to t'$, assume $\overline{\overline{M}} \models \rho$. Then there exists $\overline{\overline{\alpha}}\colon t_{\overline{\overline{M}}} \Rightarrow t'_{\overline{\overline{M}}}$. We define $\overline{\alpha} = \beta(\overline{\overline{\alpha}})$ and because β is full and surjective on objects and $\overline{\overline{\alpha}}$ is natural, we can easily prove that $\overline{\alpha}$ is natural too.

Directly from the second part of Corollary 5.6 we deduce the following:

Corollary 5.8 *If $\boldsymbol{B}^2_h \to \boldsymbol{B}_h$ is full for each $h \in H$, then for each model M and each sentence $[\rho]$,*

$$M \models\!\equiv [\rho] \quad \textit{iff} \quad M \models\!\equiv^2 [\rho]$$

Corollary 5.9 *If the system of data D is a partial order, then for each model M and each sentence $[\rho]$*

$$M \models\!\equiv [\rho] \quad \textit{iff} \quad M \models\!\equiv^2 [\rho]$$

6 Conclusions and Future Research

We internalised behavioural satisfaction to RWL by using a semantic defintion of behavioural satisfaction between hidden sorted rewrite models on one side, and equations and rules on the other side. This gives an instituion for HSRWL which generalises both HSA and RWL, thus providing a unitary logic underlying the integration of the two specification paradigms.

This paper concentrates on the semantic and logical foundations of this new paradigm, but also makes some links to the proof theory. We feel that further work needs to be done in the following areas:

- exploration of advanced techniques for proving behavioural properties of concurrent distributed systems by shifting to the HSRWL the techniques already developed for HSA, such as the "coinduction" of [14],
- exploration of relevant examples in connection to the above,
- clarify the relationship between the two different treatments in HSRWL of the object paradigm inherited from HSA and RWL, and
- further study of the relationship between behavioural satisfaction and 2-behavioural satisfaction.

We also plan to use HSRWL as a framework for defining the formal semantics of the CafeOBJ system [9] which actaully implements both specification paradigms. In fact CafeOBJ was the first driving force behind this research.

A An example

We devote this appendix for gradually illustrating the concepts introduced in this paper by using the (simple) example of a behavioural specification in HSRWL of a non-deterministic counter (the code is written in CafeOBJ [9]).

The data
As data we consider the natural numbers with non-deterministic choice as specified by the following RWL module importing a library of natural numbers.

```
module NAT? {
  extending (NAT)
  op _?_ : Nat Nat -> Nat
  vars M N : Nat
  rule M ? N => M .
  rule M ? N => N .
}
```

This defines a "visible" signature (V, Ψ). For the model D of data we consider $|D_{\texttt{Nat}}| = \mathcal{P}_f\omega$, where $\mathcal{P}_f\omega$ is the set of all finite sets of natural numbers. $D_{\texttt{Nat}}(S, S') = \emptyset$ if $S' \not\subseteq S$ and $D_{\texttt{Nat}}(S, S') = \{S \to S'\}$ if $S' \subseteq S$. We define the interpretation of the "choice" operator ? as $S?S' = S \cup S'$ and the interpretations of the usual operations on the natural numbers in a straighforward manner, for example

$$S + S' = \{x + x' \mid x \in S, x' \in S'\} \quad \text{for all} \quad S, S' \in \mathcal{P}_f\omega$$

Notice that the interpretations of the operations are monotonic wrt inclusion between sets of natural numbers, so they are functors.

The counter: signature and models
Now we can use the data module NAT? for specifying the counter object.

```
module COUNTER {
  protecting (NAT?)
  [ Counter ]                          -- class (hidden sort)
  op add : Nat Counter -> Counter      -- method
  op read : Counter -> Nat             -- attribute
  var N : Nat
  var C : Counter
  eq read(add(N,C)) = N + read(C)  . -- structural equation
}
```

This defines a hidden sorted rewrite signature (H, Σ, E) over (V, Ψ, D) where H consists of only one hidden sort (`Counter`) and Σ contains one method (`add`) and one attribute (`read`), and E contains one structural equation.

We consider two models for the module COUNTER. The first one, denoted A, is a "history" model that interprets $A_{\texttt{Counter}}$ as the *set* $(\mathcal{P}_f\omega)^*$ of lists of finite sets of natural numbers. In other words in this model the states of the counter are implemented as the lists $[S_1 \ldots S_k]$ with $S_i \subseteq \omega$ finite for $i \in \overline{1,k}$. This is a static implementation in the sense that there are no transitions between the states of the counter. $\texttt{add}_A$ just adds new sets to the history list, and $\texttt{read}_A$ evaluates the state by $\texttt{read}_A([S_1 \ldots S_k]) = S_1 + \ldots + S_k$.

The second model, denoted $\overline{A}$ (which we will later see that is in fact the bahaviour image of the model A), implements $\overline{A}_{\texttt{Counter}}$ as $D_{\texttt{Nat}}$, $\texttt{add}_{\overline{A}}(S, C) = S + C$, and $\texttt{read}_{\overline{A}}(C) = C$. Notice that both models satisfy the structural equation.

The counter: behaviour signature and systems; final behaviour system
The hidden sorted signature (H, Σ) determines a behaviour signature C where $|C| = \{\texttt{Counter}, d\}$. Because we have only one method and one (unparameterised by data) attribute, C is the category freely generated by the set of arrows $C[\texttt{Counter}, \texttt{Counter}] = \mathcal{P}_f\omega$ and $C[\texttt{Counter}, d]$ having only one element. This means that both $C(\texttt{Counter}, \texttt{Counter})$ and $C(\texttt{Counter}, d)$ consists of lists of finite sets of natural numbers with list concatenation as arrow composition in C, i.e., $C(\texttt{Counter}, \texttt{Counter}) = C(\texttt{Counter}, d) = (\mathcal{P}_f\omega)^*$.

The behaviour system $[A]$ interprets $\texttt{Counter}$ as $A_{\texttt{Counter}}$ and

$$[A]_{[S_1 \ldots S_k] : \texttt{Counter} \to \texttt{Counter}}([S'_1 \ldots S'_m]) = [S_1 \ldots S_k S'_1 \ldots S'_m]$$

$$[A]_{[S_1 \ldots S_k] : \texttt{Counter} \to d}([S'_1 \ldots S'_m]) = S_1 + \ldots + S_k + S'_1 + \ldots + S'_m$$

while $[\overline{A}]$ interprets $\texttt{Counter}$ as $\overline{A}_{\texttt{Counter}} = D_{\texttt{Nat}}$ and

$$[\overline{A}]_{[S_1 \ldots S_k]}(S) = S_1 + \ldots + S_k + S$$

in both cases. This definition can be extended on arrows by using the monotonicity of $+$ wrt the inclusions between finite sets of natural numbers, this making $[\overline{A}]_{[S_1 \ldots S_k]}$ functors.

The final behaviour system $\boldsymbol{B}_\Sigma$ interprets $\boldsymbol{B}_{\texttt{Counter}}$ as $D^{(\mathcal{P}_f\omega)^*}$.

The counter: behaviour image and satisfaction
It is easy to calculate the definition of the component $q_{\texttt{Counter}}$ of the unique behaviour system morphism $q\colon [A] \to \boldsymbol{B}$:

$$q_{\texttt{Counter}}([S_1 \ldots S_k])([S'_1 \ldots S'_m]) = S_1 + \ldots + S_k + S'_1 + \ldots + S'_m$$

Notice that $q_{\texttt{Counter}} : (\mathcal{P}_f\omega)^* \to D^{(\mathcal{P}_f\omega)^*}$ is *not* full because $A_{\texttt{Counter}}$ is a discrete category (i.e., a set). It is easy to notice that

$$[A] \to [\overline{A}] \to \boldsymbol{B}$$

is a factorisation for q, therefore $\overline{A}$ is the behaviour image of A. Because of the factorisation system in $\mathbb{C}at$, $\overline{A}_{\texttt{Counter}} \to \boldsymbol{B}_{\texttt{Counter}}$ is full, therefore the behaviour image $\overline{A}$ also adds transitions apart of the usual identification between behavioural equivalent states.

Consider the rule

$$[\texttt{add(M?N, C)}] \Rightarrow [\texttt{add(M, C)}]$$

It is clear that A *does not* satisfy this rule in the ordinary RWL sense because the model A does not implement any transitions for the counter. However, $A \models\!\!\!\equiv [\texttt{add(M?N, C)}] \texttt{=>} [\texttt{add(M, C)}]$ because $\overline{A} \models [\texttt{add(M?N, C)}] \texttt{=>} [\texttt{add(M, C)}]$.

We can actually prove this rule by using context induction, i.e., proving that

$$\texttt{COUNTER} \vdash c(\texttt{add(M?N, C)}) \Rightarrow c(\texttt{add(M, C)})$$

for each context of visible sort. The only context of length 1 is $\texttt{read}(z)$, so $\texttt{read(add(M?N, C))} \texttt{=>} \texttt{read(add(M, C))}$ follows by an application of the structural equation on both sides and by one application of the congruence rule for $\texttt{M?N} \texttt{=>} \texttt{M}$. Each context c of length $n+1$ is of the form $\texttt{read}(\texttt{add}(S, c'))$ where $\texttt{read}(c'(z))$ is a context of length n. By the structural axiom this reduces to $S + \texttt{read}(c'(\texttt{add(M?N, C)})) = \texttt{read}(c'(\texttt{add(M, C)}))$ which can be proved by the induction hypothesis and by one application of the rule of congruence.

References

[1] Michael Arbib and Ernest Manes. *Arrows, Structures and Functors.* Academic, 1975.

[2] Francis Borceaux. *Handbook of Categorical Algebra*, volume 2. Cambridge University Press, 1994.

[3] Rod Burstall and Răzvan Diaconescu. Hiding and behaviour: an institutional approach. In A. William Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, pages 75–92. Prentice-Hall, 1994. Also in Technical Report ECS-LFCS-8892-253, Laboratory for Foundations of Computer Science, University of Edinburgh, 1992.

[4] Răzvan Diaconescu. *Category-based Semantics for Equational and Constraint Logic Programming.* PhD thesis, University of Oxford, 1994.

[5] Răzvan Diaconescu. Completeness of category-based equational deduction. *Mathematical Structures in Computer Science*, 5(1):9–41, 1995.

[6] Răzvan Diaconescu. A category-based equational logic semantics to constraint programming. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Recent Trends in Data Type Specification*, volume 1130 of *Lecture Notes in Computer Science*, pages 200–221. Springer, 1996. Proceedings of 11th Workshop on Specification of Abstract Data Types. Oslo, Norway, September 1995.

[7] Răzvan Diaconescu. Category-based modularisation for equational logic programming. *Acta Informatica*, 33(5):477–510, 1996. To appear.

[8] Răzvan Diaconescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge, 1993. Proceedings of a Workshop held in Edinburgh, Scotland, May 1991.

[9] Kokichi Futatsugi and Toshimi Sawada. Design considerations for cafe specification environment. In *Proc. OBJ2 10th Anniversary Workshop*, October 1995.

[10] Joseph Goguen. Types as theories. In George Michael Reed, Andrew William Roscoe, and Ralph F. Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford, 1991. Proceedings of a Conference held at Oxford, June 1989.

[11] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.

[12] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Harmut Ehrig and Fernando Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 1–34. Springer, 1994.

[13] Joseph Goguen and Răzvan Diaconescu. An introduction to category-based equational logic. In V.S. Alagar and Maurice Nivat, editors, *Algebraic Methodology and Software Technology*, volume 936 of *Lecture Notes in Computer Science*, pages 91–126. Springer, 1995.

[14] Joseph Goguen and Grant Malcolm. A hidden agenda, 1996. draft.

[15] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen, editor, *Algebraic Specification with OBJ: An Introduction with Case Studies.* Cambridge, to appear 1995. Also to appear as Technical Report from SRI International.

[16] Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 1971.

[17] F. William Lawvere. Functorial semantics of algebraic theories. *Proceedings, National Academy of Sciences, U.S.A.*, 50:869–872, 1963. Summary of Ph.D. Thesis, Columbia University.

[18] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, (93):73–155, 1992.