An Interface Algebra for the Relational Nets

Grégory Mermoud, Marc A. Schaub, and Grégory Théoduloz

School of Computer and Communication Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL),
1015 Lausanne, Switzerland

Abstract. We present two interface algebras with their corresponding implementation relation that are interface theories for the relational nets. Toghether with a formal and thorough description of these interface algebras, we provide a proof of this claim. Moreover, we also accentuate non-triviality and expressiveness of our solution.

1 Introduction

A block diagram is used for depicting systems structure and consists of entities called blocks that communicate with each others on the basis of interconnection topology. A formalism for it, on which we rely in this paper, is given in [1]. Each block may represent a system entity at two levels: (1) the *component* itself or (2) the component *interface*. An implementation relation links these two representations. Moreover, an algebra may be built for each level of abstraction in order to reason formally on system structure. Interface algebras support compositional design (i.e. the split of a given interface F in k blocks F_1, F_2, ..., F_k) while component algebra do not. On the other hand, component algebras support compositional verification (i.e. the merging of k components f'_1, f'_2, ..., f'_k satisfying a corresponding specification f still satisfies f) while interface algebras do not. The implementation relation links the component and interface algebras in order to apply both compositional design and verification on the same system, even though such algebras are not always easy to find.

[1] introduces a particular component algebra called *relational nets* that corresponds to a generic system in computer science: sets of atomic blocks called processes which are connected by channels. Together with the relational nets, the authors provide several interface algebras that are implemented by a subset of relational nets and thus provide an interface theory for each of them. On the other hand, an interface theory for the whole set of relational nets is missing. Herein we propose two such interface algebras. Both are non-trivial and one has the important property of expressiveness.

Since this paper is tightly related to [1], we assume all definitions and claims established in it and re-use the same notations whenever possible.

2 Original Problem

2.1 Problem Statement

Problem 1. Find an interface algebra with an implementation relation that is an interface theory for the relational nets.

In addition to that requirement, we want the interface algebra to be non-trivial and as useful as possible. Defining formally what non-triviality and usefulness mean is almost impossible. Nevertheless, some obvious properties should be verified by sensible solutions. For instance, the hierarchy should allow different levels of abstraction and the connection and composition should not be empty relations. Regarding the implementation, it is required in the definition that every interface must have at least an implementing component; it is sensible to require the converse as well: for every component, an interface should exist. Note that all these requirements are fulfilled by the interfaces defined in [1].

2.2 Results

It is crucial for a relational net to have a satisfiable relation. In particular, this property should be kept when composing and connecting components together. In order to be able to ensure that at least one consistent valuation exists, it is possible to build an interface algebra that requires from implementing components that some specified valuations are consistent with the component. These required valuations would guarantee that the composition and connection are possible. Such an algebra in defined hereafter.

Formally, a *stateless RV interface* F is a stateless I/O interface $\Pi_F = (I_F, O_F^+, O_F)$ and a set of *required valuations* $R_F \subseteq [P_F]$. R_F cannot be an empty set except if P_F is itself empty. Intuitively, $v \in R_F$ means that the implementation must at least allow the valuation v. The original idea we presented in class is a subset of RV interfaces with exactly one valuation per interface.

`Composition` $F \parallel G$ is defined iff $\Pi_{F\parallel G} = \Pi_F \parallel \Pi_G$ is defined. Then $R_{F\parallel G} = \{v \in [P_{F\parallel G}] \mid (\exists\, v_F \in R_F, v_G \in R_G \mid v = v_F \uplus v_G\}$

`Connection` $F\theta$ is defined iff the result $(\Pi_{F\theta}, R_{F\theta})$, as defined next, is a stateless RV interface. Let $\Pi_{F\theta} = \Pi_F\theta$ and $v_{F\theta} \in R_{F\theta}$ iff $(\exists v_F \in R_F, \forall x \in P_F \mid v_{F\theta}(x) = v_F(x)) \wedge (\forall (x,y) \in \theta \mid v_{F\theta}(x) = v_{F\theta}(y))$.

`Hierarchy` $F' \preceq F$ iff (1) $\Pi_{F'} \preceq \Pi_F$ and (2) $(\forall v_F \in R_F, \exists v_{F'} \in R_{F'}, \forall x \in P_F \mid v_F(x) = v_{F'}(x))$.

Proposition 1. *The stateless RV interfaces are an interface algebra*

Proof: First we have to show that stateless RV interfaces are a block algebra. This is trivially the case since we rely on stateless I/O interfaces, which themselves are a block algebra, and our additions respect the properties of a block algebra, in particular commutativity and associativity of composition and reflexivity and transitivity of hierarchy.

We now need to show that stateless RV interfaces satisfy the two properties defining an interface algebra:

(1) If $F \parallel G$ is defined and $F' \preceq F$, then $\Pi_F \parallel \Pi_G$ is defined (by definition of composition for stateless RV interfaces) and $\Pi_{F'} \preceq \Pi_F$ (by definition of hierarchy for stateless RV interfaces). Given that Π_F, $\Pi_{F'}$ and Π_G are stateless I/Os, which are an interface algebra (Proposition 1 in [1]), $\Pi_{F'} \parallel \Pi_G$ is defined and $\Pi_{F'} \parallel \Pi_G \preceq \Pi_F \parallel \Pi_G$ (part 1 of the definition of an interface algebra), which means that $\Pi_{F'\parallel G} \preceq \Pi_{F\parallel G}$.

We consider an arbitrary $\widehat{v} \in R_{F\parallel G}$. By definition of the composition for stateless RV interfaces, $\widehat{v} = \widehat{v}_F \uplus \widehat{v}_G$ with $\widehat{v}_F \in R_F, \widehat{v}_G \in R_G$. Since $F' \preceq F$, there is $\widehat{v}_{F'} \in R_{F'}$ such that $(\forall x \in P_F | \widehat{v}_F(x) = \widehat{v}_{F'}(x))$. We now consider $\widehat{v'} = \widehat{v}_{F'} \uplus \widehat{v}_G \in R_{F'\parallel G}$ and a given $\widehat{x} \in P_{F\parallel G}$. By definition of $v_F \uplus v_G$, if $\widehat{x} \in P_F$, then $\widehat{v}(x) = \widehat{v}_F(x)$ and $\widehat{v}_{F'}(x) = \widehat{v}'(x)$, which means that $\widehat{v}(x) = \widehat{v}'(x)$. Similarly, if $\widehat{x} \in P_G$, then $\widehat{v}(x) = \widehat{v}_G(x) = \widehat{v}'(x)$, and therefore $(\forall x \in P_{F\parallel G} \mid v_{F\parallel G}(x) = v_{F'\parallel G}(x))$.

Since we also have $\Pi_{F'\parallel G} \preceq \Pi_{F\parallel G}$, we can conclude that $F' \parallel G \preceq F \parallel G$. Furthermore, $\Pi_{F'} \parallel \Pi_G$ being defined implies that $F' \parallel G$ is defined, which means that stateless RV interfaces satisfy the first condition for being an interface algebra.

(2) We assume that $F\theta$ is defined and $F' \preceq F$. Let's consider an arbitrary $\widehat{v}_{F\theta} \in R_{F\theta}$. By definition of connection, we have $\widehat{v}_F \in R_F$ such that $(\forall x \in P_F \mid \widehat{v}_{F\theta}(x) = \widehat{v}_F(x))$, and by definition of hierarchy we have $\widehat{v}_{F'} \in R_{F'}$ such that $(\forall x \in P_F \mid \widehat{v}_F(x) = \widehat{v}_{F'}(x))$. We can now consider $\widehat{v}_{F'\theta}$ such that $(\forall x \in P_F \mid \widehat{v}_{F'\theta}(x) = \widehat{v}_{F'}(x) = \widehat{v}_F(x) = \widehat{v}_{F\theta}(x))$ and $(\forall (x,y) \in \theta \mid \widehat{v}_{F'\theta}(x) = \widehat{v}_{F'\theta}(y))$, which means that $\widehat{v}_{F'\theta} \in R_{F'\theta}$ by definition of connection. Since we also have $\Pi_{F'\theta} = \Pi_{F'}\theta$ and $\Pi_F\theta = \Pi_{F\theta}$ (by definition of connection) and $\Pi_{F'}\theta \preceq \Pi_F\theta$ (because Π_F and $\Pi_{F'}$ are stateless I/Os and $F' \preceq F \Rightarrow \Pi_{F'} \preceq \Pi_F$ by definition of hierarchy), we can conclude that $F'\theta \preceq F\theta$.

Furthermore, we can deduct from the reasoning above that if $R_{F\theta}$ is non-empty, then $R_{F'\theta}$ is non-empty too, which is sufficient to conclude that if $F\theta$ is defined, then $F'\theta$ is defined too. The stateless RV interfaces therefore also satisfy the second condition for being an interface algebra. □

We now have to introduce an implementation relation between RV interfaces and relational nets.

For a relational net f, let $\rho_f^A = \bigwedge_{a \in A_f} \rho_a$ and $\rho_f^C = \bigwedge_{(x,y) \in C_f} (x = y)$ and $\rho_f = \rho_f^A \wedge \rho_f^C$. The predicate ρ_f is true at least at the valuations in R_F. Given a relational net f and a stateless RV interface F, define $f \lhd_{RV} F$ iff (1) $f \lhd_{I/O} \Pi_F$, (2) $(\forall v_F \in R_F, \exists v_f \in [P_f] \mid \rho @ v_f \wedge (\forall x \in I_f \cup O_F \mid v_f(x) = v_F(x)))$.

Proposition 2. *The stateless RV interfaces with $\lhd_{RV}$ are an interface theory for the relational nets.*

In order to prove this statement, we need to show that $\lhd_{RV}$ is a compositional implementation of RV interfaces by relational nets. Below are the intuitions for proving the three properties of a compositional implementation. This is intuitively the case since we heavily rely on I/O interfaces and $\lhd_{I/O}$, which respects the assumptions made on ports, and we directly derive the implementation from the required valuations.

(1) For all relational nets f and g and all RV interfaces F and G, if $f \lhd F$ and $g \lhd G$ and $F \parallel G$, $f \parallel g$ is defined since $P_f \cap P_g =$ by definition of $\lhd_{I/O}$ and composition for I/O interfaces and $f \parallel g \lhd_{RV} F \parallel G$ because the required values $R_{F\parallel G}$, as defined for RV are the union of the required values R_F and R_G, which are implemented by f respectively g.

(2) $F\theta$ and $f\theta$ are defined in a very similar way. $F\theta$ is necessarily an RV interface, and since we add the same connections, if $f \lhd F$ then $f\theta \lhd F\theta$.

(3) This is trivially the case since all valuations of the abstract component F are included in the refined component F', and therefore if $f \lhd F'$, it has to include at least all valuations that are required for $f \lhd F$.

3 Refined Problem

3.1 Problem Statement

The previous section has shown that the original problem statement leads to a solution: RV interfaces. Nevertheless, RV interfaces are not completely satisfactory. Properties found in the stateless PD interfaces (that are interfaces for total nets) are not present in the suggested RV interfaces. In particular, refining a RV interface goes in a direction which is opposed to the one in PD design: a refined RV interfaces has more constraints whilst in PD, the refined version allows less connections. Therefore, PD truly goes in the other direction than the total nets while RV reflect in a too close way what is done in relational nets.

Consequently, it would be desirable to have an extended interface algebra in which whatever is possible with the PD interfaces would still be possible. [1] introduces a notion of expressiveness which perfectly captures what is desired. Using that concept, the problem can be restated, and therefore refined.

Problem 2 (refined version). Find an interface algebra with an implementation relation that is an interface theory for the relational nets *and which is as expressive as the stateless PD interfaces on the subset of the total nets.*

3.2 Results

The stateless RV interfaces clearly do not satisfy the new requirement. It is not possible to find a mapping from PD interfaces to RV interfaces that has the desired properties. Note that the two are incomparable regarding expressiveness because PD interfaces are not as expressive as RV interfaces as well.

To have an interface algebra as expressive as the stateless PD interface algebra on the total nets but that would be usable at the same time with any relational nets, a combination of the given solution to the original problem – stateless RV interfaces – and the stateless PD interfaces has been created. The intuitive idea is to have interfaces in the new algebra that are composed of both a RV and a PD part that may be connected only in some restricted ways. A schematic vision of the desired result is given in Figure 1. The idea is to allow only connections within the two blocks and from the partial part to the total one. Even though this approach may seem restrictive, it is a satisfactory compromise that allows to mix ports for which certain pairs have to be present (RV) and ports among which it is enough not to have cycles (PD). Moreover the definition of PD+RV interfaces is built so as to mainly rely on the previously defined interfaces (I/O, PD and RV) so as to reuse the previous results as much as possible.

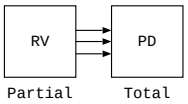

Fig. 1. Schematic vision of a RV+PD interface

Formally, a *stateless RV+PD interface* F consists of (1) a stateless RV interface $\Pi^I_F = (I^I_F, O^{+I}_F, O^I_F, R^I_F)$, (2) a stateless PD interface $\Pi^O_F = (I^O_F, O^{+O}_F, O^O_F, \kappa^O_F)$ and (3) an interconnect $\alpha_F \subseteq P^I_F \times I^O_F$, such that $P_{\Pi^I_F} \cap P_{\Pi^O_F} = \varnothing$. *The set of ports* P_F is defined as $P_{\Pi^I_F} \cup P_{\Pi^O_F}$.

Intuitively, a RV+PD interface is composed of two disjoint interfaces: one RV interface for the partial part and one PD for the total part. Ports from the partial parts can be connected to the total input ports but not the other way around. Connections from a PD port to a RV port are forbidden since we have no information on the possible outputs of a PD interface; therefore we cannot guarantee that the values that the RV interface expects as input can be realised.

Additionally, *the associated stateless I/O interface* Π^{IO}_F is defined as $(I^I_F \cup I^O_F, O^{+I}_F \cup O^{+O}_F, O^I_F \cup O^O_F)$.

`Composition` $F \parallel G$ is defined iff (1) both $\Pi^I_F \parallel \Pi^I_G$ and $\Pi^O_F \parallel \Pi^O_G$ are defined and (2) $P^+_F \cap P^+_G = \varnothing$. Then, $\Pi^I_{F\parallel G} = \Pi^I_F \parallel \Pi^I_G$, $\Pi^O_{F\parallel G} = \Pi^O_F \parallel \Pi^O_G$ and $\alpha_{F\parallel G} = \alpha_F \cup \alpha_G$.

Notation. Before defining the connection of an interface, we define some sets of ports and subsets of an interconnect that help with the definition of the

connection. Consider an interconnect θ and a RV+PD interface F. Let *the set of free input (resp. output) ports of θ relative to F, $FI_{F,\theta}$ (resp. $FO_{F,\theta}$)*, be defined as $I_{F,\theta} \setminus P_F$ (resp. $O_\theta \setminus P_F$). *The set of free ports $FP_{F,\theta}$* is defined as the union of the free input and output ports relative to F.
A connection $(x, y) \in \theta$ belongs to *the RV-internal part of the interconnect, denoted by θ^I_F*, iff $(x \in P_{\Pi^I_F} \vee y \in P_{\Pi^I_F}) \wedge (x \notin P_{\Pi^O_F} \wedge y \notin P_{\Pi^O_F})$. Similarly, a connection $(x, y) \in \theta$ belongs to *the PD-internal part of the interconnect, denoted by θ^O_F*, iff $(x \in P_{\Pi^O_F} \cup FP_{F,\theta} \vee y \in P_{\Pi^O_F} \cup FP_{F,\theta}) \wedge (x \notin P_{\Pi^I_F} \wedge y \notin P_{\Pi^I_F})$.
If a connection is neither RV-internal nor PD-internal, it said to be *external*. Intuitively, external connections are the one between the RV and PD parts. Let θ^e_F be the set of external connections. Note that θ^I_F, θ^O_F and θ^e_F is a partition of θ.

`Connection` $F\theta$ is defined iff (1) $\Pi^I_F\theta^I_F$ is defined, (2) $\Pi^O_F\theta^O_F$ is defined, (3) for all (x, y) in θ^e_F, $x \notin P_{\Pi^O_F}$ and (4) $\Pi^{IO}_F\theta$ is defined. The two first conditions ensure that there are no illegal internal connections, the third avoids "backward" connections and the fourth rules out potentially remaining inconsistencies (like an internal and an external connection with the same destination port). Then, $\Pi^I_{F\theta} = \Pi^I_F\theta^I_F$, $\Pi^O_{F\theta} = \Pi^O_F\theta^O_F$ and $\alpha_{F\theta} = \alpha_F \cup \theta^e_F$.

`Hierarchy` $F' \preceq F$ iff (1) $\Pi^I_{F'} \preceq \Pi^I_F$, (2) $\Pi^O_{F'} \preceq \Pi^O_F$ and (3) $\alpha_{F'} \cap (P_F \cap P_{F'})^2 \supseteq \alpha_F \cap (P_F \cap P_{F'})^2$. Note that this definition of the hierarchy is fairly restrictive. In particular, ports must be of the same kind (either in the RV or PD part) and the interconnect must be identical for the common ports.

Proposition 3. *The stateless RV+PD interfaces are an interface algebra*

Proof: Proving that proposition relies heavily on the fact that PD and RV are themselves interfaces. First one must prove that this is a block algebra. This first proof should be straightforward. Second one must prove that this is an interface algebra. Once again, the proof is fairly straightforward, relying systematically on what has been established for PD and RV interfaces: since the definitions of composition, connection and hierarchy on PD+RV interfaces directly rely on these definitions on PD interfaces and on RV interfaces, the fact that both PD and RV interfaces are known to be an interface algebra can be used to build a proof.

It remains to show that the stateless RV+PD interfaces with some given compositional implementation relation (1) are an interface algebra for the relational nets and (2) are as expressive as the stateless PD interfaces on the total nets.

Notation. Consider a relational net $f = (A_f, C_f)$. Let $P_f = \bigcup_{a \in A_f} P_a$. A *subnet of f* τ is a relational net (A_τ, C_τ) such that (1) $A_\tau \subseteq A_f$, (2a) for all $(x, y) \in C_\tau$, $x, y \notin P_f \setminus P_\tau$ (i.e. x and y are not ports of another process) and (2b) $C_f \cap P^2_\tau \subseteq C_\tau$ (i.e. all connections among ports are in the subnet).
A triplet (τ_1, τ_2, π), where τ_1 and τ_2 are subnets of f and π a subset of $P_f \times P_f$, is

a partition of f if (1) $A_{\tau_1} \uplus A_{\tau_2} = A_f$ (i.e. they are a partition of the processes), (2) $\pi = (C_f \cap ((P_{\tau_1} \times P_{\tau_2}) \cup (P_{\tau_2} \times P_{\tau_1}))$ and (3) $C_{\tau_1} \uplus C_{\tau_2} \uplus \pi = C_f$ (i.e. every connection of f has to be either in τ_1 or τ_2 except the ones between the two subnets).

A relational net $f = (A_f, C_f)$ implements a RV+PD interface F, written $f \triangleleft_{PD+RV} F$, iff there exists a partition of f (τ_1, τ_2, π) such that (1) $\tau_1 \triangleleft_{RV} \Pi_F^I$, (2) τ_2 is total and $\tau_2 \triangleleft_{PD} \Pi_F^I$ and (3) $\alpha_F \cap (P_f \cap P_F)^2 = \pi \cap (P_f \cap P_F)^2$. The intuition for the implementation relation is that the PD interface must be implemented by a total subnet and the RV interface by the rest of the relational net.

Proving that $\triangleleft_{PD+RV}$ is compositional can be done by essentially relying on the facts that both $\triangleleft_{PD}$ and $\triangleleft_{RV}$ are compositional.

Since a RV+PD interface "embeds" a PD interface, it seems obvious that the stateless RV+PD interfaces are as expressive as the stateless PD interfaces on the set of the total nets. The proof of that fact is straightforward considering the mapping α_{PD} from a PD interface to a corresponding RV+PD one defined as follows: for all PD interfaces F, $\alpha_{PD}(F) = (\varnothing_{RV}, F, \varnothing)$ where $\varnothing_{RV}$ is the RV interface such that $P_{\varnothing_{RV}} = \varnothing$. The proof is then straightforward: the definitions of composition, connection and hierarchy for RV+PD interfaces use the corresponding definitions for PD interfaces and the definition of the implementation relation restricted to the set of total nets relies directly on the implementation relation defined for PD interfaces.

Therefore, RV+PD interfaces is an appropriate solution to the refined problem: it is an interface theory for the relational nets that is as expressive as PD interfaces.

Moreover, RV+PD interfaces are as expressive as RV interfaces. The reason is the very same as before, considering the mapping α_{RV} which is defined as follows: for all RV interfaces F, $\alpha_{RV}(F) = (F, \varnothing_{PD}, \varnothing)$ where $\varnothing_{PD}$ is the PD interface such that $P_{\varnothing_{PD}} = \varnothing$.

Finally, the proposed solution is particularly expressive and extension to it would not allow much more interesting designs. For instance, if one wanted to allow connection from a PD port to a RV port, one would have to add some information on the possible output values of the PD interface. It would not be sufficient to give the output domain (i.e. the set (or a subset) of possible output values) because we generally cannot guarantee that the input ports on which that output port depends can always take all the possible values. Typically, if some RV output port is connected to some PD input port, we can only say that the input port will take the values given in the required valuations. Consequently, if we wanted to give a sufficient information on an PD output port so as to allow connection with a RV input port, we would need to give some output requirements (i.e. whatever the input is, the output is guaranteed to be able to take all the required values). It would be particularly constraining (considering which relational net would implement it) and it is always possible in that case to use a RV interface that would contain valuations with the output ports having the required values. This most obvious modification does not increase expressiveness significantly.

4 Conclusion

The objective of this project is really hard to reach but our results eventually fulfil the requirements. Indeed, we propose in this paper an interface algebra as expressive as PD on the set of total nets. The requirement of expressiveness is not a part of the original assignment, which has therefore been refined in order to lead to more interesting solutions.

This issue enables us to understand how carefully theoretical problems should be handled since they may lead to plausible solutions from a strictly theoretical point of view which are actually useless or not expressive enough. In fact, this illustrates that stating a problem appropriately is often as crucial as the problem itself and may radically influence the whole solving flow.

Finally, the path leading to our solution is maybe even more important than the solution itself since it showed us how many ways *seem* to lead to various solutions while it appears that only a few – if not only one – good solutions actually exist.

References

1. L. de Alfaro and T. A. Henzinger, “Interface theories for component-based design.”.