Independent Implementability of Viewpoints

Thomas A. Henzinger[1,⋆] and Dejan Ničković[2]

[1] IST Austria, Klosterneuburg, Austria
[2] AIT Austrian Institute of Technology, Vienna, Austria

Abstract. Interface theories provide a formal framework for component-based development of software and hardware which supports the incremental design of systems and the independent implementability of components. These capabilities are ensured through mathematical properties of the parallel composition operator and the refinement relation for components. More recently, a conjunction operation was added to interface theories in order to provide support for handling multiple viewpoints, requirements engineering, and component reuse. Unfortunately, the conjunction operator does not allow independent implementability in general.

In this paper, we study conditions that need to be imposed on interface models in order to enforce independent implementability with respect to conjunction. We focus on multiple viewpoint specifications and propose a new compatibility criterion between two interfaces, which we call orthogonality. We show that orthogonal interfaces can be refined separately, while preserving both orthogonality and composability with other interfaces. We illustrate the independent implementability of different viewpoints with a FIFO buffer example.

1 Introduction

Component-based design is a common design methodology, where complex systems are developed by assembling individual components. It usually involves a combination of *bottom-up* and *top-down* design techniques. In bottom-up design, the designer assembles the overall system by integrating already available components. Top-down design starts from the specification of the overall system that is decomposed and refined into requirements for the subsequent design stages.

Interface theories [6] were developed as a formal framework that supports both bottom-up and top-down approaches in component-based design. An *interface* is an abstract specification that describes the interaction of a component with its environment. In particular, an interface captures both the *assumptions* that the component makes about its environment, and the *guarantees* that the component provides when used in the intended design context. In order to support bottom-up design, interface theories provide a *composition* operator that satisfies

⋆ This work was supported in part by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling) and by the FWF National Research Network RISE (Rigorous Systems Engineering).

R. Calinescu and D. Garlan (Eds.): Monterey Workshop 2012, LNCS 7539, pp. 380–395, 2012.

the *incremental design* property. Two interfaces are *compatible* for composition if their port types match and there exists a design context in which they can interact without violating their mutual guarantees. Incremental design requires the possibility of checking the compatibility of two interfaces and composing them, without considering the precise design context in which the composition will be used. The composition operator is both *associative* and *commutative*, thus ensuring that compatible interfaces can be developed independently and composed in any order. In top-down design, the notion of *refinement* plays a central role. This design flow starts with a system-level interface that is iteratively decomposed and refined into sub-system interfaces, until implementations of respective components are obtained. Top-down design, illustrated in Figure 1, is subject to the *independent implementability* property, that requires the possibility of refining compatible interfaces separately, while still maintaining compatibility between them.

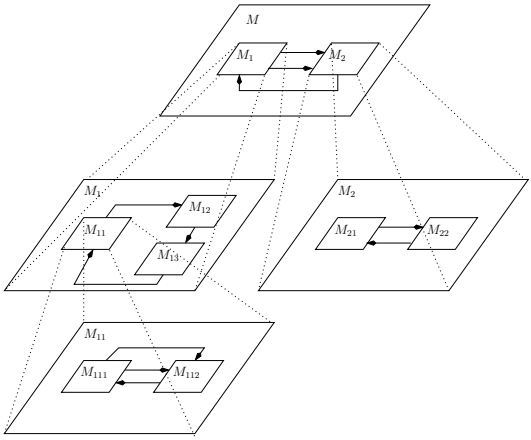

Fig. 1. Top-down design

The properties of the composition operator and the refinement relation, provide necessary basic support for bottom-up and top-level design in interface theories. However, composition alone does not cover all the aspects that are encountered in component-based design, such as:

1. Specification of component's *multiple viewpoints*, where each viewpoint is modeled as an interface, and specifies a particular (behavioral, timing, power consumption, etc.) aspect of the component;
2. *Requirement engineering*, by formal modeling of requirement documents that consist of a conjunction of individual requirements;
3. *Component reuse* in different parts of a design.

In order to provide additional support for the above aspects of component based design, the *conjunction* operator was introduced in [8], in the context of stateless and Moore interfaces [4]. The conjunction is a partial function defined on pairs of interfaces and is the most general refinement of individual interfaces, i.e. the greatest lower bound in the refinement lattice on interfaces. The conjunction between two interfaces is defined if they are *consistent*[1], i.e. if their input variables do not overlap with the output variables and the output guarantees do not contradict each other. The conjunction operator was subsequently added to modal interfaces [10], assume/guarantee contracts [2] and synchronous relational interfaces [12]. Top-down design with conjunction is illustrated in Figure 2 in the context of multiple viewpoints and component reuse, where components can be reused in different parts of a design, without being restricted to be trees of components, but they can also be directed acyclic graphs.

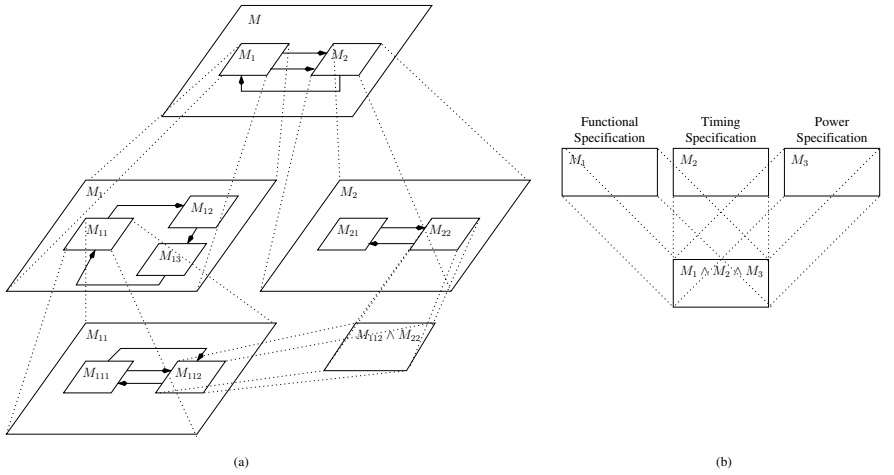

Fig. 2. Top-down design with conjunction for: (a) component reuse, and (b) multiple viewpoints

The conjunction of specifications plays an important role in top-down design. A natural requirement for the conjunction operator would be to support independent implementability and allow separate stepwise refinement of individual interfaces. Unfortunately, conjunction does not satisfy the independent implementability property, in general. We illustrate this point with the following example.

Example 1. Let M, M' and N be three stateless interfaces shown in Figure 3. The input variables of M and N do not overlap with their respective output variables. Furthermore, the output guarantees of M and N do not contradict, i.e. there always exists an output $(y = 2)$, such that both the guarantees of M

[1] In [8], consistency is called shared refinability.

and N are satisfied. It follows that M and N are consistent. Moreover, M' refines M because the two interfaces accept the same inputs and the output guarantee of M' implies the output guarantee of M. However, M' and N are not consistent, because the conjunction of their output guarantees is unsatisfiable, hence the conjunction of M' and N is not defined. It follows that M and N cannot be refined independently, given that the consistency property is not preserved by refinement.

$$M : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y \leq 2 \end{cases} \quad M' : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y < 2 \end{cases} \quad N : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y \geq 2 \end{cases}$$

Fig. 3. Single-state interfaces M, M' and N: M and N are consistent, M' refines M, but M' and N are not consistent

In this paper, we study sufficient conditions that need to be imposed on stateless and Moore interfaces, in order to guarantee their independent implementability with respect to the conjunction operator. We focus in particular on the context where conjunction is used to combine multiple viewpoints of the same component. Independent implementability of viewpoints is a highly desirable requirement of an interface theory, because different aspects of a component are often specified and developed by different design teams, and are not effectively combined until a late stage in the design process. We first observe that different viewpoints of a component usually specify non-overlapping aspects, and the guarantees that are provided by individual viewpoints are rarely conflicting. It follows that the notion of consistency is not well adapted to conjunction of viewpoints. We instead propose a different criterion, that we call *orthogonality*, for conjunction of two interfaces to be defined. We say that two interfaces are orthogonal if their input variables do not overlap with their output variables, and if the intersection of their output variables is empty. While this condition is not realistic for specifying multiple requirements of the same view of a component, we believe it is reasonable for expressing multiple view requirements. We show that for every two interfaces that are orthogonal, they can be refined separately, while maintaining orthogonality between them.

2 Stateless Interfaces

A stateless interface consists of a set of input and output variables, an input assumption predicate and an output guarantee predicate.

Definition 1 (Stateless interface). *A* stateless interface $M = \langle X^I, X^O, \varphi, \psi \rangle$ *consists of the following components:*

- X^I *and* X^O *are disjoint sets of* input *and* output *variables. We define* $X = X^I \cup X^O$*;*
- φ *is a predicate over* X^I *called* input assumption*; and*
- ψ *is a predicate over* X^O *called* output guarantee*.*

We require the stateless interface to be *well-formed*, i.e. to accept at least one input value and generate at least one output value.

2.1 Connection, Composition and Refinement

In this section, we define standard connection and parallel composition operators, as well as the refinement relation, as in [7], and recall the incremental design and independent implementability properties that are supported by stateless interfaces.

A connection consists of a set of interface variable pairs and defines which variables in an interface are interconnected after application of the connection operator. For all pairs, the first component is an output and the second component an input variable of the stateless interface to which the output is connected. Formally, we have the following:

Definition 2 (Connection). *A* connection θ *is a set of pairs* (x, y)*, consisting of a* source *variable* x *and a target variable* y*, such that for all pairs* $(x, y), (x', y') \in \theta$*, if* $x \neq x'$*, then* $y \neq y'$*.*

We denote by $\mathcal{S}_\theta$ the set of source variables in θ, by $\mathcal{T}_\theta$ the set of target variables in θ, and by ρ_θ the predicate $\bigwedge_{(x,y)\in\theta}(x = y)$.

We say that a connection θ is *compatible* with a stateless interface M, if the following conditions hold: (1) the source variables in θ are all output variables of M; (2) the target variables in θ are all input variables in M; and (3) when source variables are connected to target variables, there exists a valuation of remaining input variables in M for which the assumption φ of M is satisfied for all values of output variables of M that satisfy the guarantee ψ of M.

Definition 3 (Compatibility for connection). *A stateless interface* $M = \langle X^I, X^O, \varphi, \psi \rangle$ *is* compatible *with a connection* θ *if the following conditions hold:*

- $\mathcal{S}_\theta \subseteq X^O$*;*
- $\mathcal{T}_\theta \subseteq X^I$*;*
- *the predicate* $\hat{\varphi} = \forall X^O . \forall \mathcal{T}_\theta . ((\psi \wedge \rho_\theta) \rightarrow \varphi)$ *is satisfiable.*

Given an interface M and a connection θ such that M is compatible with θ, the result of applying θ to M is the stateless interface $M\theta = \langle \hat{X}^I, \hat{X}^O, \hat{\varphi}, \hat{\psi} \rangle$, where

- $\hat{X}^I = X^I \backslash \mathcal{T}_\theta$;
- $\hat{X}^O = X^O \cup \mathcal{T}_\theta$; and
- $\hat{\psi} = (\psi \wedge \rho_\theta)$

Example 2. The application of a connection θ to a stateless interface M is illustrated in Figure 4. In this example, $\theta = \{(z, x)\}$ and the predicate $\hat{\varphi} = (\forall x, z)((z < 2 \wedge z = x) \rightarrow (x < 3 \wedge y \neq 0))$ is satisfiable and can be simplified to $y \neq 0$.

$$M : \begin{cases} \text{var: } \begin{cases} \text{in: } x, y \ : \ \mathbb{N} \\ \text{out: } z \ : \ \mathbb{N} \end{cases} \\ \text{A: } x < 3 \wedge y \neq 0 \\ \text{G: } z < 2 \end{cases} \quad M\theta : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{N} \\ \text{out: } y, z \ : \ \mathbb{N} \end{cases} \\ \text{A: } y \neq 0 \\ \text{G: } z < 2 \wedge x = z \end{cases}$$

Fig. 4. Stateless interfaces M and $M\theta$, where $\theta = \{(z, x)\}$

Theorem 1 ([7]). *Let M be a well-formed stateless interface and θ be a connection. If M is compatible with θ, then $M\theta$ is a well-formed stateless interface.*

Parallel composition operator supports combination of compatible stateless interfaces. We say that two stateless interfaces are compatible for parallel composition if (1) their output variables are disjoint; (2) the input variables of each stateless interface are disjoint of the output variables of the other stateless interface; and (3) the conjunction of their guaranteed is satisfiable.

Definition 4 (Compatibility for composition). *Two stateless interfaces $M = \langle X^I_M, X^O_M, \varphi_M, \psi_M \rangle$ and $N = \langle X^I_N, X^O_N, \varphi_N, \psi_N \rangle$ are compatible for parallel composition if*

- $X^O_M \cap X^O_N = \emptyset$;
- $X^I_M \cap X^O_N = X^I_N \cap X^O_M = \emptyset$; *and*
- $\varphi_M \wedge \varphi_N$ *is satisfiable.*

Formally, parallel composition of two compatible stateless interfaces is defined as follows:

Definition 5 (Parallel composition). *Given two stateless interfaces $M = \langle X^I_M, X^O_M, \varphi_M, \psi_M \rangle$ and $N = \langle X^I_N, X^O_N, \varphi_N, \psi_N \rangle$ which are compatible for parallel composition, their parallel composition is the interface $M \mid\mid N = \langle \hat{X}^I, \hat{X}^O, \hat{\varphi}, \hat{\psi} \rangle$, where*

- $\hat{X}^I = X^I_M \cup X^I_N$;
- $\hat{X}^O = X^O_M \cup X^O_N$;
- $\hat{\varphi} = (\varphi_M \wedge \varphi_N)$; *and*
- $\hat{\psi} = (\psi_M \wedge \psi_N)$.

Theorem 2 ([7]). *Let M and N be two well-formed stateless interfaces. If M and N are compatible for parallel composition, then $M \parallel N$ is a well-formed stateless interface.*

We say that a stateless interface N *refines* the stateless interface M if it has more permissive assumption and more restrictive guarantees.

Definition 6 (Refinement). *Given two well-formed stateless interfaces $M = \langle X_M^I, X_M^O, \varphi_M, \psi_M \rangle$ and $N = \langle X_N^I, X_N^O, \varphi_N, \psi_N \rangle$, we say that N refines M, denoted by $N \preceq M$, if*

- $(X_M^I \cup X_N^I) \cap (X_M^O \cup X_N^O) = \emptyset$;
- $\varphi_M \rightarrow \varphi_N$ *is valid; and*
- $\psi_N \rightarrow \psi_M$ *is valid.*

Following definitions of refinement, compatibility for connection, connection, compatibility for composition and parallel composition, we have that stateless interfaces satisfy the independent implementability with respect to both connection and composition, as stated in the following theorem.

Theorem 3 ([7]). *Let M and N be two well-formed stateless interfaces and θ be a connection. If $N \preceq M$ and M is compatible with θ, then N is compatible with θ and $N\theta \preceq M\theta$.*

Let M, N and S be three well-formed stateless interfaces such that $X_N \cap X_S \subseteq X_M$. If M and S are compatible for composition and $N \preceq M$, then N and S are compatible for composition and $N \parallel S \preceq M \parallel S$.

2.2 Conjunction

The conjunction $M \wedge N$ of two stateless interfaces M and N was introduced in [8] as an interface meant to work in two environments based on separate descriptions of each environment. The interface $M \wedge N$ allows inputs that satisfy assumptions of either M or N, and provides the guarantees of both M and N. In order to ensure that the conjunction of two interfaces is well-formed, the notion of consistency was introduced. Two stateless interfaces are said to be consistent if (1) the input variables do not overlap with the output variables and (2) their output guarantees do not contradict each other.

Definition 7 (Conjunction). *Given two consistent stateless interfaces $M = \langle X_M^I, X_M^O, \varphi_M, \psi_M \rangle$ and $N = \langle X_N^I, X_N^O, \varphi_N, \psi_N \rangle$, the conjunction of M and N is the stateless interface $M \wedge N = \langle \hat{X}^I, \hat{X}^O, \hat{\varphi}, \hat{\psi} \rangle$, where*

- $\hat{X}^I = X_M^I \cup X_N^I$;
- $\hat{X}^O = X_M^O \cup X_N^O$;
- $\hat{\varphi} = (\varphi_M \vee \varphi_N)$; *and*
- $\hat{\psi} = (\varphi_M \wedge \varphi_N)$.

Theorem 4 ([8]). *Let M and N be two well-formed stateless interfaces. If M and N are consistent, then $M \wedge N$ is a well-formed stateless interface.*

The conjunction of two stateless interfaces subsumes all behaviors of the given interfaces, as stated in the following theorem.

Theorem 5 ([8]). *Let M and N be two well-formed stateless interfaces. If M and N are consistent, then $M \wedge N \preceq M$ and $M \wedge N \preceq N$, and for all well-formed stateless interfaces S, if $S \preceq M$ and $S \preceq N$, then $S \preceq M \wedge N$.*

Unfortunately, conjunction does not support independent implementability of stateless interfaces, as demonstrated in Figure 3. The reason comes from the consistency condition between two stateless interfaces, that is not preserved by refinement. Given consistent stateless interfaces M and N, the output guarantees of M and N do not conflict by definition. However, another interface M' that refines M may strengthen its guarantees in a way that makes the output guarantees of M' and N conflicting. Thus, stateless interfaces M' and N may not be consistent. However, we observe that in the case that the conjunction operator is used to combine multiple viewpoints of the same component, the output variables of the individual viewpoints are usually disjoint, hence their output guarantees cannot contradict each other. In fact, the consistency between two stateless interfaces is not preserved by refinement. Following this observation, we propose a new condition between two stateless interfaces, that we call *orthogonality*.

Definition 8 (Orthogonality). *Let M and N be two well-formed stateless interfaces. We say that M and N are* orthogonal *if $(X_M^I \cup X_N^I) \cap (X_M^O \cup X_N^O) = \emptyset$ and $X_M^O \cap X_N^O = \emptyset$.*

We believe that in the context of multiple viewpoint specifications, the orthogonality is a realistic requirement. Note that while the output variables of two orthogonal interfaces are disjoint, the two interfaces are interacting with each other through common input variables. In the following lemma, we show that orthogonal interfaces are consistent.

Lemma 1. *Let M and N be two well-formed stateless interfaces. If M and N are orthogonal, then M and N are consistent.*

Proof. Assume that M and N are well-formed and orthogonal. It follows that both ψ_M and ψ_N are satisfiable. By definition, we have that ψ_M is a predicate over X_M^O and ψ_N is a predicate over X_N^O, and by assumption we have that $X_M^O \cap X_N^O = \emptyset$. It follows that there exists a valuation over $X_M^O \cup X_N^O$ that satisfies both ψ_M and ψ_N, hence $\hat{\psi}$ is also satisfiable.

□

Following Lemma 1, we are ready to state the result that establishes the independent implementatibility property for the conjunction operator between orthogonal stateless interfaces.

Theorem 6 (Independent implementability of conjunction). *Let M, N and S be three well-formed stateless interfaces such that $X_N^O \cap X_S = X_N \cap X_S^O = \emptyset$. If M and S are orthogonal and $N \preceq M$, then N and S are orthogonal and $N \wedge S \preceq M \wedge S$.*

Proof. Assume that M and S are orthogonal and $N \preceq M$. By Lemma 1, M and S are consistent, hence $M \wedge S$ is defined.

We have by definition of a stateless interface that (1) $X_N^I \cap X_N^O = X_S^I \cap X_S^O = \emptyset$. By the assumption that $X_N^O \cap X_S = \emptyset$ and $X_N \cap X_S^O = \emptyset$, we have that (2) $X_N^I \cap X_S^O = X_N^O \cap X_S^I = \emptyset$. By (1) and (2), it follows that $(X_N^I \cup X_S^I) \cap (X_N^O \cap X_S^O) = \emptyset$.

Furthermore, by the assumption that $X_N^O \cap X_S = \emptyset$, we have that $X_N^O \cap X_S^O = \emptyset$. It follows that N and S are orthogonal, hence by Lemma 1 consistent, and $N \wedge S$ is defined.

By the assumption, we have that (3) $N \preceq M$ and by Theorem 5, we have that (4) $N \wedge S \preceq N$. By (3) and (4), we have that (5) $N \wedge S \preceq M$. By Theorem 5, we have that (6) $N \wedge S \preceq S$. Finally, by (5), (6) and Theorem 5, we can conclude that $N \wedge S \preceq M \wedge S$.

□

Example 3. Consider stateless interfaces M and N shown in Figure 5. The two interfaces do not share output variables, hence they are orthogonal. Stateless interface N' refines N, by constraining its guarantee predicate. It is not hard to see that the conjunction $M \wedge N'$ refines $M \wedge N$, illustrating the independent implementability property of the conjunction operator.

$$M : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y \leq 2 \end{cases} \quad N : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } z \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } z \geq 0 \end{cases} \quad M \wedge N : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y, z \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y \geq 2 \wedge z \geq 0 \end{cases}$$

$$N' : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } z \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } z \bmod 2 = 0 \end{cases} \quad M \wedge N' : \begin{cases} \text{var: } \begin{cases} \text{in: } x \ : \ \mathbb{B} \\ \text{out: } y, z \ : \ \mathbb{N} \end{cases} \\ \text{A: } \textsc{True} \\ \text{G: } y < 2 \wedge z \bmod 2 = 0 \end{cases}$$

Fig. 5. Stateless interfaces M, N and N': M and N are orthogonal, and $N' \preceq N$, hence M and N' are orthogonal and $M \wedge N' \preceq M \wedge N$

3 Moore Interfaces

In this section, we consider *Moore interfaces*, a synchronous interface model, that was first introduced in [4]. Moore interfaces have internal states, that are

decorated with assumption predicates over input variables, and guarantee predicates over output variables. We consider Moore interfaces with deterministic transition relation, where transitions are guarded by predicates over input and output variables of the interface.

Definition 9 (Moore interface). *A* Moore interface $M = \langle X^I, X^O, Q, \hat{q}, \varphi, \psi, \rho \rangle$ *consists of the following components:*

- X^I *and* X^O *are disjoint sets of* input *and* output *variables. We define* $X = X^I \cup X^O$*;*
- Q *is a finite set of* locations*, and* $\hat{q} \in Q$ *is the* initial *location;*
- φ *is a* labeling *that associates with each location* $q \in Q$ *an* input assumption predicate *over* X^I*;*
- ψ *is a* labeling *that associates with each location* $q \in Q$ *an* output guarantee predicate *over* X^O*;*
- ρ *is a* transition guard *that associates with each pair of locations* $q, q' \in Q$ *a predicate* $\rho(q, q')$ *over* X*.*

Given a set X of variables, a *valuation* v over X is a function that assigns to each $x \in X$, a value $v(x)$ of the appropriate type. We denote by $\mathcal{V}[X]$, the set of all valuations v over X. Given a predicate φ on X, we write $v \models \varphi$ if the valuation v satisfies φ.

An *execution* of M is a sequence $q_0, v_0, q_1, \ldots, q_n, v_n, q_{n+1}$ of states $q_i \in Q$ and valuations $v_i \in \mathcal{V}[X]$ such that: (1) $q_0 = \hat{q}$ is the initial state of M, and (2) $v_i \models \varphi(q_i) \wedge \psi(q_i) \wedge \rho(q_i, q_{i+1})$. We say that the sequence $v_0, \ldots, v_n$ is the *trace* of M, and that the states $q_0, \ldots, q_{n+1}$ are *reachable* in M.

The Moore interfaces can in general be non-deterministic, or even block in some executions. Thus, we consider only *well-formed* interfaces, where the well-formedness criterion is defined as follows:

Definition 10 (Well-formedness). *A Moore interface* $M = \langle X^I, X^O, Q, \hat{q}, \varphi, \psi, \rho \rangle$ *is* well-formed *if for all states* q *that are reachable in* M*: (1) both* $\varphi(q)$ *and* $\psi(q)$ *are satisfiable; (2)* $(\varphi(q) \wedge \psi(q)) \rightarrow \exists q'.\ \rho(q, q')$ *is valid, and (3)* $((\rho(q, q') \wedge (\rho(q, q'')))) \rightarrow q' = q''$ *is valid for all* $q', q'' \in Q$*.*

Well-formedness ensures that the interface is non-blocking by conditions (1) and (2), and deterministic by (3).

3.1 Composition and Refinement

In this section, we define standard parallel composition operator and refinement relation in the lines of [4], and recall the incremental design and independent implementability properties that are supported by Moore interfaces.

The parallel composition is a partial function on pairs of Moore interfaces, that is defined if the two interfaces are compatible. We say that two interfaces are compatible if their variable types match and if there exists a design context in which the two interfaces can interact in a way that preserves their individual guarantees.

Definition 11 (Compatibility and parallel composition). *Given two Moore interfaces* $M = \langle X_M^I, X_M^O, Q_M, \hat{q}_M, \varphi_M, \psi_M, \rho_M \rangle$ *and* $N = \langle X_N^I, X_N^O, Q_N, \hat{q}_N, \varphi_N, \psi_N, \rho_N \rangle$*, let* $X^O = X_M^O \cup X_N^O$*,* $X^I = (X_M^I \cup X_N^I) \backslash X^O$*,* $Q = Q_M \times Q_N$*,* $\hat{q} = (\hat{q}_M, \hat{q}_N)$*, and for all* $q, q' \in Q_M$ *and* $r, r' \in Q_N$*,* $\psi(q, r) = \psi_M(q) \wedge \psi_N(r)$ *and* $\rho((q, q'), (r, r')) = \rho_M(q, q') \wedge \rho_N(r, r')$*. We say that* M *and* N *are* compatible*, if* $X_M^O \cap X_N^O = \emptyset$*, and there exists a labeling* $\varphi_\otimes$ *such that for all executions* $(q_0, r_0), v_0, \ldots, v_{n-1}, (q_n, r_n)$ *of* $\langle X^I, X^O, Q, \hat{q}, \varphi_\otimes, \psi, \rho \rangle$*, we have that* $v_i \models (\varphi_M(q_i) \wedge \varphi_N(r_i))$ *for all* $0 \leq i \leq n$*.*

The parallel composition $P = M \mid\mid N$ *is defined if and only if* M *and* N *are compatible, in which case* $P = \langle X^I, X^O, Q, q_0, \varphi, \psi, \rho \rangle$*, where* φ *is the* weakest labeling *that satisfies the above conditions.*

The parallel composition operator is associative, thus supporting incremental design, i.e. ensuring that the compatible interfaces of a system can be put together in any order.

Theorem 7 ([4]). *Given three Moore interfaces* M*,* N *and* S*, either* $M \mid\mid (N \mid\mid S)$ *and* $(M \mid\mid N) \mid\mid S$ *are both undefined, or they are both defined and equal.*

The refinement of two Moore interfaces is defined as an alternating simulation relation R, and we say that R is a *witness* for $N \preceq M$.

Definition 12 (Refinement). *Given two Moore interfaces* $M = \langle X_M^I, X_M^O, Q_M, \hat{q}_M, \varphi_M, \psi_M, \rho_M \rangle$ *and* $N = \langle X_N^I, X_N^O, Q_N, \hat{q}_N, \varphi_N, \psi_N, \rho_N \rangle$*, we say that* N refines M*, denoted by* $N \preceq M$*, if*

1. $(X_M^I \cup X_N^I) \cap (X_M^O \cup X_N^O) = \emptyset$*, and*
2. *there exists a relation* $R \subseteq Q_M \times Q_N$ *such that: (1)* $(\hat{q}_M, \hat{q}_N) \in R$*,* $\varphi_M(q) \rightarrow \varphi_N(r)$ *is valid; (2)* $\psi_N(r) \rightarrow \psi_M(q)$ *is valid, and (3) for all* $q' \in Q_M$ *and* $r' \in Q_N$*, if* $\varphi_M(q) \wedge \psi_N(r) \wedge \rho_M(q, q') \wedge \rho_N(r, r')$ *is satisfiable, then* $(q', r') \in R$*.*

Following definitions of refinement, compatibility and composition, it follows that Moore interfaces satisfy the independent implementability requirement, as stated in the following theorem:

Theorem 8 ([4]). *Let* M*,* N *and* S *be three well-formed Moore interfaces such that* $X_N \cap X_S \subseteq X_M$*. If* M *and* S *are compatible, and* $N \preceq M$*, then* N *and* S *are also compatible and* $N \mid\mid S \preceq M \mid\mid S$*.*

3.2 Conjunction

The conjunction $M \wedge N$ of two Moore interfaces M and N was also introduced and defined in [8] as the weakest interface that refines both M and N, similarly to the stateless case. In the context of a conjunction of Moore interfaces M and N, as long as the inputs satisfy both the assumptions of M and N, the outputs must satisfy both the guarantees of M and N. If the assumption of M (N) is

violated, then the conjunction interface does not need anymore to satisfy the guarantees of M (N) and jumps to a copy of N (M). The conjunction does not allow inputs that violate both assumptions of M and N.

Definition 13 (Conjunction). *Given two Moore interfaces* $M = \langle X^I_M, X^O_M, Q_M, \hat{q}_M, \varphi_M, \psi_M, \rho_M \rangle$ *and* $N = \langle X^I_N, X^O_N, Q_N, \hat{q}_N, \varphi_N, \psi_N, \rho_N \rangle$*, let* P *be the Moore interface* $\langle X^I, X^O, Q, \hat{q}, \varphi, \psi, \rho \rangle$*, where*

- $X^I = X^I_M \cup X^I_N$
- $X^O = X^O_M \cup X^O_N$
- $Q = (Q_M \times Q_N) \cup Q_M \cup Q_N$
- $\hat{q} = (\hat{q}_M, \hat{q}_N)$
- φ *and* ψ *are defined for all* $q \in Q_M$ *and* $r \in Q_N$*, by*

$$\begin{array}{llll} \varphi(q,r) = (\varphi_M(q) \vee \varphi_N(r)) & & \psi(q,r) = (\psi_M(q) \wedge \psi_N(r)) & \\ \varphi(q) \quad = \varphi_M(q) & & \psi(q) \quad = \psi_M(q) & \\ \varphi(r) \quad = \varphi_N(r) & & \psi(r) \quad = \psi_N(r) & \end{array}$$

- ρ *is defined for all* $q, q' \in Q_M$ *and* $r, r' \in Q_N$*, by*

$$\begin{array}{ll} \rho((q,r),(q',r')) & = (\varphi_M(q) \wedge \varphi_N(r) \wedge \rho_M(q,q') \wedge \rho_N(r,r')) \\ \rho((q,r),q') & = (\varphi_M(q) \wedge \neg\varphi_N(r) \wedge \rho_M(q,q')) \\ \rho((q,r),r') & = (\neg\varphi_M(q) \wedge \varphi_N(r) \wedge \rho_N(r,r')) \\ \rho(q,q') & = \rho_M(q,q') \\ \rho(r,r') & = \rho_N(r,r') \\ \rho(q,(q',r')) & = \rho(r,(q',r')) = \bot \end{array}$$

We say that M *and* N *are consistent if: (1)* $X^I \cap X^O = \emptyset$*, and (2)* $\psi(q)$ *is satisfiable for all states* q *that are reachable in* $M \wedge N$*.*

When M *and* N *are consistent, the conjunction* $M \wedge N$ *is the well-formed Moore interface* P*.*

Theorem 9 ([8]). *Let* M *and* N *be two well-formed Moore interfaces. If* M *and* N *are consistent, then* $M \wedge N \preceq M$ *and* $M \wedge N \preceq N$*, and for all well-formed Moore interfaces* S*, if* $S \preceq M$ *and* $S \preceq N$*, then* $S \preceq M \wedge N$*.*

In Section 1, we have seen that the conjunction operator does not support independent implementability in general, and hence similarly to the stateless interface case, we introduce the same orthogonality condition for Moore interfaces.

Definition 14 (Orthogonality). *Let* M *and* N *be two well-formed Moore interfaces. We say that* M *and* N *are* orthogonal*, if* $(X^I_M \cup X^I_N) \cap (X^O_M \cup X^O_N) = \emptyset$ *and* $X^O_M \cap X^O_N = \emptyset$*.*

In the following lemma, we show that two orthogonal Moore interfaces are also consistent:

Lemma 2. *Let* M *and* N *be two well-formed Moore interfaces. If* M *and* N *are orthogonal, then* M *and* N *are consistent.*

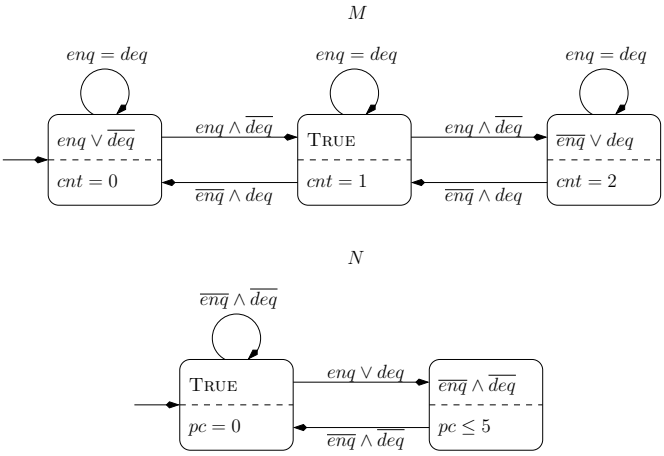

Fig. 6. FIFO buffer - functional specification M and functional/power consumption specification N

Proof. The proof is identical to the one of Lemma 1. □

Following Lemma 2, we are ready to state the result that establishes the independent implementability property for the conjunction operator between orthogonal Moore interfaces.

Theorem 10 (Independent implementability of conjunction). *Let M, N and S be three well-formed interfaces such that $X_N^O \cap X_S = X_N \cap X_S^O = \emptyset$. If M and S are orthogonal and $N \preceq M$, then N and S are orthogonal and $N \wedge S \preceq M \wedge S$.*

Proof. The proof follows the same line as the proof of Theorem 6. □

4 FIFO Buffer Example

We illustrate independent implementability of viewpoints with a FIFO buffer example. The FIFO buffer specification consists of two interfaces, M and N that describe two different aspects of the buffer. These two specifications are extensions of the example presented in [8], and are depicted in Figure 6.

The interface M specifies a buffer of size 2. M has two Boolean input variables *enq* and *deq*, that model the enqueue and dequeue operations and one integer variable *cnt* that gives the current number of items that are stored in the buffer. The assumption (guarantee) predicates are depicted in the upper (lower) part of locations, and transitions are labeled by their guards. Initially, the buffer is empty, hence $cnt = 0$. Every exclusive enqueue (dequeue) operation increases (decreases) the *cnt* variable by one. However, in the initial state, the buffer is not allowed to dequeue, and in the state where the buffer is full ($cnt = 2$), the

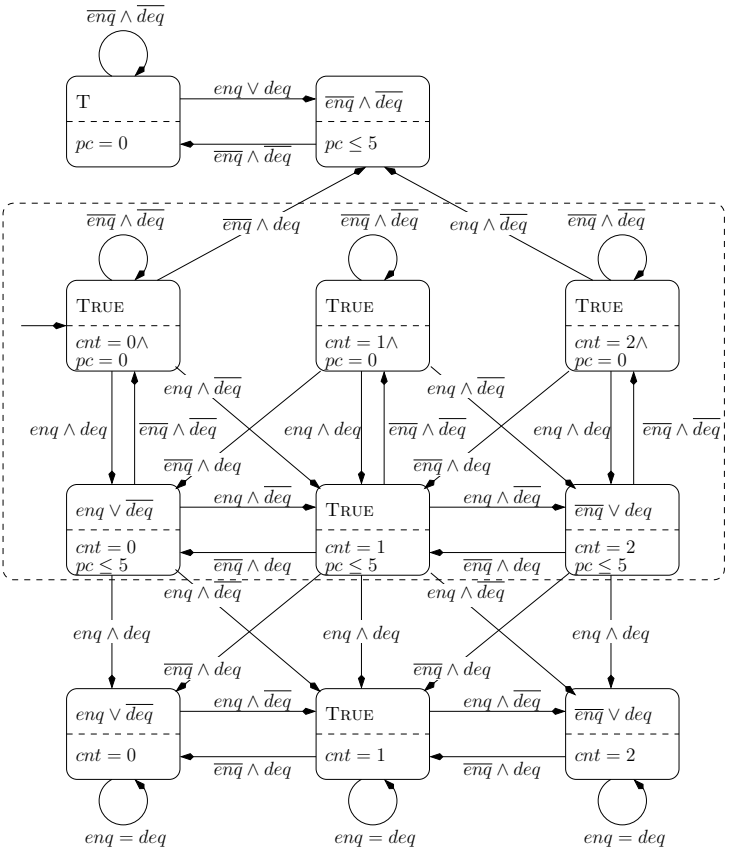

Fig. 7. FIFO buffer - conjunction $M \wedge N$

buffer is not allowed to enqueue new items. Note that simultaneous enqueue and dequeue operations are allowed, but have no effect.

Interface N specifies a power consumption and another behavioral aspect of a FIFO buffer. It has the same input variables as N (*enq* and *deq*), and an integer output variable *pc* that models the power consumption of a buffer. This interface forbids two consecutive enqueue or dequeue operations to happen. Additionally, it specifies the power consumption needed to process enqueue and dequeue requests. We can see that the absence of enqueue/dequeue requests results in no power consumption. On the other hand, any combination of the presence of enqueue and dequeue operations is bounded by 5 power units.

The interfaces M and N are naturally combined by the conjunction operator. The two interfaces are consistent, hence their conjunction $M \wedge N$ is defined and is shown in Figure 7. To obtain the conjunction $M \wedge N$, we need additional transitions leaving the dashed line box when the assumptions of M (N) are violated, and from then on only assumptions and guarantees of N (M) need to be satisfied.

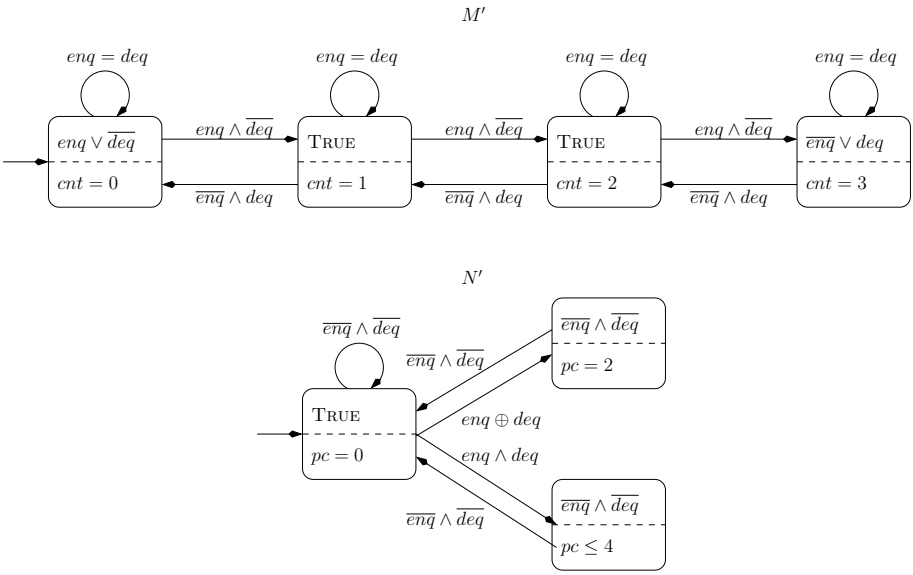

Fig. 8. FIFO buffer - functional specification M' and functional/power consumption specification N' such that $M' \preceq M$ and $N' \preceq N$

.

Interfaces M and N may be under-specified for many applications. For example, a designer may need to use a buffer that can store more than two items. Moreover, the interface N does not distinguish between the power consumption of enqueue and dequeue operations, that may have different resource requirements to be processed. Hence, M and N may need to be refined. Refining the requirements directly on the conjunction $M \wedge N$ can be highly impractical, given the relative complexity of $M \wedge N$ with respect to M and N. This can be appreciated by comparing Figures 6 and 7.

In this example, interfaces M and N are not only consistent, but also orthogonal. In fact, their output variables are disjoint, that is the number of items in the buffer does not depend on its power consumption, and vice versa. The orthogonality of M and N allows us to postpone taking an explicit conjunction of M and N, and to refine them separately and independently. In Figure 8, we show two interfaces, M' and N', where M' refines the interface M, and N' refines the interface N. M' specifies a buffer that can store up to three elements. It provides the same guarantees as M, while the number of items in the buffer is bounded by two, but is also able to process an additional enqueue request and store a maximum of three items.

On the other hand, the interface N' refines the power consumption guarantees, by distinguishing between the presence of an exclusive enqueue or dequeue request, that consumes exactly 2 power units, and a simultaneous enqueue/dequeue request that requires up to 4 power units. We leave to the reader the exercise to checking that M' and N' are indeed orthogonal and that $M' \wedge N'$ refines $M \wedge N$.

5 Conclusion

In this paper, we proposed orthogonality as a new condition between two Moore interfaces that ensures the independent implementability with respect to the conjunction operator. We believe that the orthogonality is the right notion when considering conjunction of multiple viewpoints. We demonstrated the stepwise refinement property of orthogonal interfaces with an example of a FIFO buffer that combines behavioral and power consumption specifications.

The power consumption specification of the FIFO buffer corresponds to a pure threshold resource interface in [5], but it was encoded as a Moore interface in our example. We believe that the next important step would be to study conjunction of fully heterogeneous interface models, a problem closely related to heterogeneous composition [1,3]. In particular, we are interested in models where the non-functional properties are expressed as Büchi threshold, pure energy and reward energy interfaces from [5] or real-time interfaces [9, 11].

References

1. Benveniste, A., Caillaud, B., Carloni, L.P., Caspi, P., Sangiovanni-Vincentelli, A.L.: Composing heterogeneous reactive systems. ACM Trans. Embed. Comput. Syst. 7, 43:1–43:36 (2008)
2. Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C.: Multiple Viewpoint Contract-Based Specification and Design. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2007. LNCS, vol. 5382, pp. 200–225. Springer, Heidelberg (2008)
3. Caspi, P., Benveniste, A., Lublinerman, R., Tripakis, S.: Actors without Directors: A Kahnian View of Heterogeneous Systems. In: Majumdar, R., Tabuada, P. (eds.) HSCC 2009. LNCS, vol. 5469, pp. 46–60. Springer, Heidelberg (2009)
4. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Mang, F.Y.C.: Synchronous and Bidirectional Component Interfaces. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 414–427. Springer, Heidelberg (2002)
5. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource Interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT 2003. LNCS, vol. 2855, pp. 117–133. Springer, Heidelberg (2003)
6. de Alfaro, L., Henzinger, T.A.: Interface automata. In: ESEC / SIGSOFT FSE, pp. 109–120 (2001)
7. de Alfaro, L., Henzinger, T.A.: Interface Theories for Component-Based Design. In: Henzinger, T.A., Kirsch, C.M. (eds.) EMSOFT 2001. LNCS, vol. 2211, pp. 148–165. Springer, Heidelberg (2001)
8. Doyen, L., Henzinger, T.A., Jobstmann, B., Petrov, T.: Interface theories with component reuse. In: EMSOFT, pp. 79–88 (2008)
9. Henzinger, T.A., Matic, S.: An interface algebra for real-time components. In: IEEE Real Time Technology and Applications Symposium, pp. 253–266 (2006)
10. Raclet, J.-B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. Inform. 108(1-2), 119–149 (2011)
11. Thiele, L., Wandeler, E., Stoimenov, N.: Real-time interfaces for composing real-time systems. In: EMSOFT, pp. 34–43 (2006)
12. Tripakis, S., Lickly, B., Henzinger, T.A., Lee, E.A.: A theory of synchronous relational interfaces. ACM Trans. Program. Lang. Syst. 33(4), 14 (2011)