Modal I/O Automata for Interface and Product Line Theories

Kim G. Larsen[1], Ulrik Nyman[1], and Andrzej Wąsowski[2,1]

[1] Department of Computer Science, Aalborg University
[2] Computational Logic and Algorithms Group, IT University of Copenhagen
{kgl,ulrik,wasowski}@cs.aau.dk

Abstract. Alfaro and Henzinger use alternating simulation in a two player game as a refinement for interface automata [1]. We show that interface automata correspond to a subset of modal transition systems of Larsen and Thomsen [2], on which alternating simulation coincides with modal refinement. As a consequence a more expressive interface theory may be built, by a simple generalization from interface automata to modal automata. We define modal I/O automata, an extension of interface automata with modality. Our interface theory that follows can express liveness properties, disallowing trivial implementations of interfaces, a problem that exists for theories build around simulation preorders. In order to further exemplify the usefulness of modal I/O automata, we construct a behavioral variability theory for product line development.

1 Introduction

An interface theory [1, 3–7] is a type-system-like theory for component languages, where types (*interfaces*) describe components (*implementations*) with *composition* being the only operator available. A type error proves that either a component does not *conform* to its interface, or that two composed components are *incompatible*. Since the overall structure of these type systems is so simple, it is often accepted not to give typing rules explicitly when describing interface theories (for example [1, 3–6]), focusing instead on the essential ingredients of conformance, compatibility and composition.

Regular, non-component types are only applied to existing objects in program code. In contrast for interface theories it makes sense to discuss interfaces as specifications of application's architecture in isolation from actual source code. An interface abstracts the component in terms of the assumptions made by the component and the guarantees that it provides. One reasons about possible connections between component implementations (*compositions*) by using properties of composition of interfaces; most importantly *independent implementability* (that any implementations conforming to compatible interfaces are compatible) and generality properties (that the composition of interfaces produces an interface with the weakest assumptions and strongest guarantees).

We consider behavioral interface theories suitable for specification of communication protocols between components (web services or embedded systems). Such theories typically require a *contravariant* treatment of inputs and outputs to ensure deadlock-free implementations: inputs guaranteed by the specification are always offered by the implementation and that the implementation never produces more outputs than the specification. This observation led de Alfaro, Henzinger and colleagues [1, 3, 4] to a conclusion that game theoretical models of interaction are most suitable as building blocks for behavioral interface theories. While we do appreciate the values of the game theoretical formulations, we disagree with some claims in the above cited work and argue that game formulations are insufficient in themselves: there is a genuine value in combining the game theoretical approach with more traditional formulations based on transition systems, or more precisely on modal transition systems.

The two worlds of game models and modal transition systems convey largely orthogonal information about the moves of a system. Game models specify who has *control* over transitions, while modal transition systems focus on requirements, *modality*: which moves are allowed and which are required. In this paper we try to relate the two worlds, explain their weaknesses and their qualities. Eventually we combine them into a unified interface theory.

Game theoretical notions of conformance are often based on alternating simulation [8]. We show that alternating simulation in a two player setting, as used in interface automata [1, 9], is just a special case of modal transition systems refinement developed by Larsen and Thomsen [2] in the late eighties. This suggests that the real value of the game theoretic approach to component theories does not lie in the use of alternating simulation, but in the use of *control* information in the composition synthesis algorithms.

Not surprisingly then, modal transition systems themselves cannot be used to build an interface theory, without adding control information. We build a new interface theory around *modal I/O automata,* which combine features of both game theoretic models and modal transition systems. Thanks to this new combination, our interfaces are now able to express liveness properties, which was impossible in existing interface theories (after this work has been completed we have learned about [10], which achieves a similar effect in a different setting).

In order to further demonstrate the usefulness of our modal I/O automata, we construct a *product line* [11–13] *theory.* In simple words a product line is a set of similar products built by combining *assets* from a common platform available in the development process. The differences between the products are referred to as *variability.* Our theory is a behavioral formalism for describing the variability of components. The theory supports deciding whether given requirements can be satisfied by choosing concrete instances from the set of available assets. This theory, though very small, is to the best of our knowledge one of the very few attempts at describing software product lines in a behavioral fashion, and unlike the previous work [14], which takes a top-down approach to describing product families, it facilitates a bottom up construction of products, which is how product line development is more typically understood in the software engineering community. This contribution is not meant to be comprehensive, highly developed and well set in the tradition of the product line development. It should be

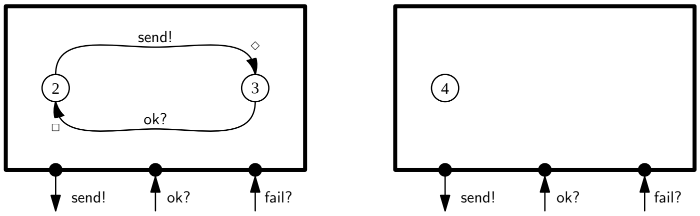

Fig. 1. The *Client* interface (left) and a trivial implementation of it (right).

understood as a simple example that emphasizes the semantic difference between modeling components in component based development and modeling assets for product family development. We do hope to extend this theory soon and report about it separately in detail.

The paper proceeds as follows. In the next section we shall explain the main results of the paper in nontechnical terms. Our main results concentrate in sections 3, 5 and 6. In Section 3 we draw a correspondence between the alternating simulation and observational modal refinement. In Section 4 modal I/O automata are defined, which are then used to construct an interface theory in Section 5 and a product line theory in Section 6. Sections 5 and 6 are largely independent, though they share a lot of intuitions. We conclude in Section 7.

2 Interface Automata vs Modal Automata: An Example

Consider an example interface automaton for a *Client* component (Fig. 1 (left), originally presented in [1]). This simple model describes a component that occasionally may want to send a package, and once it has made the request it is ready to receive an acknowledgment. The signature of the interface also mentions a fail input, but the component is never able to receive it. This means that *Client* is only capable of interacting with network links that never fail.

In interface automata, due to a game theoretic semantics, all outputs are controlled by the component itself (called the *Output* player), while all inputs to such components are controlled by the environment player (called the *Input* player). An implementation conforms to the interface iff whenever some input is offered by the interface, then it is also offered by the implementation, and whenever an implementation produces any output, this output is also present in the interface (conformance formalized as alternating simulation [8]).

Such a notion of conformance implies that compatibility can be passed from interfaces to components: if there is no winning strategy for the input player that leads to a deadlock in the interface automaton, then there won't be such a strategy for the same player that interacts directly with any implementation. Similarly if there is no strategy for the output player that leads to an output that cannot be accepted by the environment, then there is also no such strategy for any of the implementations.

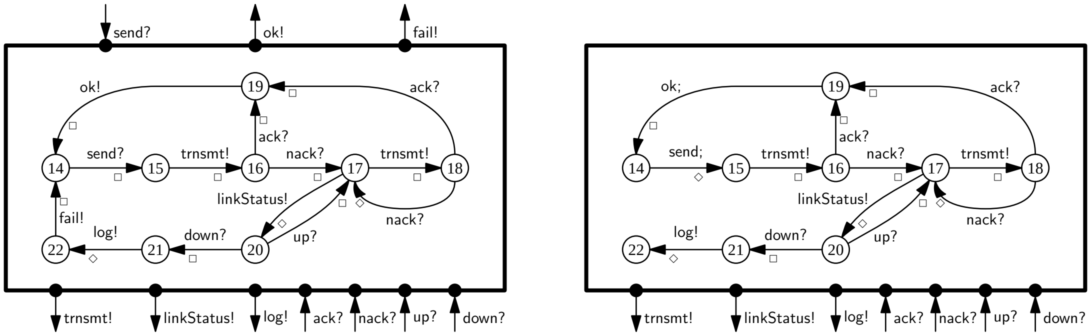

Fig. 2. *DataLink* layer with nontrivial modalities (left). Composition *DataLink* ⊗ *Client* (right). State 22 is an error state, where *DataLink* can produce the fail action, not accepted by *Client*.

Unfortunately this notion of conformance, though very much safety oriented, does not enforce that the implementations take on any useful activities at all. Consider for example the diagram on the right side of Fig. 1. It presents a model of an implementation that does not perform any actions ever. In other words this is a network application that does not use the network at all. Still this new model conforms to its interface on the left, as in its initial state it does not add any illegal outputs and it offers all the inputs that were offered by the interface.

If we turn this into the terminology used in modal transition systems it means that all the inputs are *required*, which is indicated by the $\Box$ (must) modality on the corresponding transition, and the outputs are *allowed*, which is indicated by the $\Diamond$ (may) modality on the transitions. In a modal transition systems perspective, conformance is based on modal refinement [2]. This refinement requires that whenever an implementation makes a step, then it must be possible to mimic it by an allowed transition of the specification; whenever the specification makes a required step it must be possible to match it with some required step of the corresponding state in the implementation. With the assignment of *may* to output transitions and *must* to input transitions this sounds nearly like the alternating simulation described above. In Section 3 we prove that indeed the two relations coincide if we require that the may transition relation is input-enabled.

Consequently modality gives strictly more modeling power than alternating refinement. Various modalities can be assigned to actions regardless of whom controls them. Instead of allowing all possible extensions on inputs, as in interface automata, the designer is able to control what extensions are allowed. For example we can change the *Client* model of Fig. 1 to have a must modality ($\Box$) on the *send!* transition, which will have the effect that now all the implementations must be able to proceed producing an output. This would rule out trivial implementations as the one presented on the right side of Fig. 1.

The game theoretic formulation of conformance gives a certain interpretation to inputs and outputs. Namely that inputs are *incoming requests* for service (for example remote procedure calls), while outputs are *outgoing requests* for

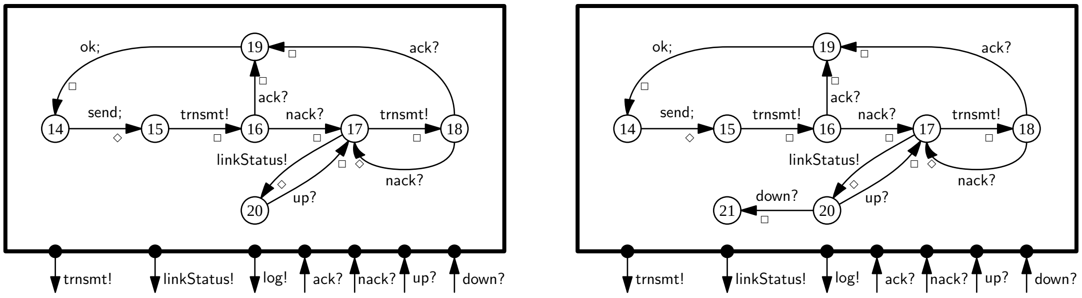

Fig. 3. Composed interfaces *LinkLayer* | *Client* and variability models *LinkLayer* · *Client*

service (also remote procedure calls, albeit in the other direction). With such an interpretation it becomes clear that removing services from the promised list should be illegal, while removing calls to external services is perfectly fine. This is exactly what alternating simulation achieves. What it misses is a more complex structure of communication.

In asynchronous systems some messages indeed convey calls for service, however many other return feedback from the services (return a value). When a given output models returning a value from a component, then clearly it should never be removed, as then the whole component becomes useless. Fig. 2 illustrates another interface modeling a data link layer, which exploits the interplay between control and modality. The *must* modality is placed on transmt! transitions, as the data link layer would be useless if the implementation was permitted not to forward packets down the stack. Similarly the transition sending back the error message cannot legally be removed. At the same time the call for linkStatus! is a may transition as some implementations are allowed not to consult the hardware link explicitly to detect errors. Finally not all implementations are forced to be able to work with links that fail twice in a row, which is modeled by the second nack! transition being a may transition.

Now consider how the two interfaces of Fig. 1 (left) and Fig. 2 (left) should be composed. The composition resembles a product computation (taken separately for the may transition relation and the must transition relation). As a result we obtain the interface presented on the right side of Fig. 2. Because the client component was so weak, the ultimate interface shows a system that possibly may never do anything. However if *Client* will send some packets, these packets will certainly be processed by the composition, unless the hardware link is broken. In such a case it might be that the implementation will produce a fail! message which will cause a deadlock with the current version of the *Client* (this can happen when the composition is in state 22). Since we cannot modify the composed system we instead synthesize a new interface which restricts the use of the composition in order to guarantee error freeness. States of the composition that can experience deadlocks are called *error states*. We follow Alfaro and Henzinger in removing error states, and transitively all states from which error states can be reached by following *internally controllable transitions* of the component (outputs and internal actions). This leads to the interface on Fig. 3 (left), expressing the fact that this component works well as long as the physical link never goes down.

The pruning mechanism described above would not be possible without the information describing which transitions are internally controllable being explicitly present in the model. It does not seem possible to compute the safe fragment of the product automaton, by just investigating the modalities of transitions. While we have said that modal refinement is strictly more expressive than alternating simulation, the control information of interface automata has its unique qualities too: it enables valuable synthesis algorithms not otherwise possible.

Let us now revisit the model of Fig. 2 (left) giving it a different interpretation than previously. Instead of perceiving it as an abstraction of a component, we should now see it as a description of a set of components. A modal automaton describes in fact a whole, often infinite, set of possible implementation automata[3]. One can think of them as all possible configurations of the model. This feature of modal automata suggests the possibility of using them as a behavioral formalism in describing variability in product lines.

A product line is a collection of products that are similar in that they offer overlapping functionality, and in that they are built from assets selected from a common platform. In here we want to describe both assets and the whole product line by modal I/O automata. If each of the assets is modeled as a modal I/O automaton we can model the capabilities of the family by composing these descriptions. However this time we would not be interested in a composition that guarantees compatible behavior of any selection of assets. It is normally expected that not all the assets in a product line platform are mutually compatible. Some of them will deadlock (for example a failing link layer and our *Client* component). The requirement for composing the variability descriptions is not to synthesize an interface that guarantees correctness of composition of all possible combination of assets, but to precisely describes what the correct combinations are: i.e. what are the deadlock free behaviors respecting the modalities that can be constructed with the available automata.

It turns out that a composition like that exists and it resembles the pruning of the product automaton for interface automata. The only difference is that now error states are the states where the error must be possible to realize (so one party must be required to produce an output that the other party must not be allowed to receive) and that we prune all the states from which reaching an error state is unavoidable (in our interface theory we have pruned states from which reaching errors might be possible).

The result of composing *Client* and *LinkLayer* using the variability model semantics is presented on the right side of Figure 3. This result contains a slightly

[3] This is also true for interface automata, though to a much lesser extent. Due to the lack of modality the set of implementations for an interface automaton is much simpler than it can be for a modal automaton.

bigger model than the interface automaton composition on the left. It states that there exists a pair of assets (implementations of *Client* and *LinkLayer*) such that it is able to accept a link down message without an error message. The transition with the down message was removed in the interface compositions as, for some pairs of implementations, it would lead to a deadlock.

Can a given specification be implemented by choosing components from available assets? Is the result of the composition the most general possible, containing all possible legal products? Can we find what the configuration of these elements should be? We address some of these questions in section 6, with an intention of elaborating more in upcoming work.

3 Alternating Simulation vs Modal Refinement

Let us begin with defining modal automata, a version of modal transition systems [2] extended with signatures. A modal automaton has two transition relations indicating respectively allowed (*may*) and required (*must*) behavior.

Definition 1 (Modal Automaton). *A modal automaton S is a six tuple: $S = (states_S, start_S, ext_S, int_S, \longrightarrow_{\diamond}, \longrightarrow_{\Box})$ where $states_S$ is a finite set of states, $start_S \in states_S$ is the initial state, ext_S and int_S are disjoint sets of external and internal actions and $act_S = ext_S \cup int_S$, $\longrightarrow_{\diamond S} \subseteq states_S \times act_S \times states_S$ is the may transition relation describing allowed behavior, and $\longrightarrow_{\Box S} \subseteq states_S \times act_S \times states_S$ is the must transition relation describing required behavior.*

Throughout the paper we sometimes use the symbols “!”, “?” and “;” after an action. This is done in order to increase the readers intuition of whether the action is respectively an output, input or internal action. No symbol is used when the action can be of more than one type. These symbols could be left out completely as it is the identity of the action that is significant.

In the following we write $s \xrightarrow{\mathcal{I}}{}^{*}_{\Box} s'$ meaning that there exists a sequence of internal *must* actions leading from s to s'. The same is defined for *may* transitions.

A modal automaton is *syntactically consistent* if everything that is required is also allowed, such that $\longrightarrow_{\Box} \subseteq \longrightarrow_{\diamond}$. In the following we only consider syntactically consistent modal automata. A modal automaton is an *implementation* if the two transition relations coincide.

A modal automaton describes a set of possible implementations. Simplistically when refining a modal automaton specification into an implementation one can remove a *may* transition, that does not have a corresponding *must* transitions or strengthen it into a *must* transition. In general this refinement is not syntactic, but behavioral, so it is not the syntactic transitions that are refined but the actual steps taken by the transition system. The same transition can be refined differently each time it is taken.

Definition 2 (Modal Refinement). *For a pair of modal automata S and T with the same signature, a binary relation $R \subseteq states_S \times states_T$ is a modal refinement if whenever sRt and $a \in act_S$ it holds that*

if $t \xrightarrow{a}_{\Box} t'$ *then* $\exists s'. s \xrightarrow{a}_{\Box} s'$ *and* $(s', t') \in R$.
if $s \xrightarrow{a}_{\Diamond} s'$ *then* $\exists t'. t \xrightarrow{a}_{\Diamond} t'$ *and* $(s', t') \in R$.

Modal refinement $\leq_{\mathrm{m}}$*is defined as the largest such relation. We say that a modal automaton S modally refines a modal automaton T, written $S \leq_{\mathrm{m}} T$, iff there exists a modal refinement containing $(start_S, start_T)$.*

Observational modal refinement is a weaker refinement in which the two modal automata can take internal transitions, that cannot be directly observed by the other automaton. In absence of internal actions the observational refinement coincides with the non-observational one.

Definition 3 (Observational Modal Refinement). *For a pair of modal automata S and T with the same signature, a binary relation $R \subseteq states_S \times states_T$ is an observational modal refinement if whenever sRt and $a \in act_S$ it holds that*

if $t \xrightarrow{a}_{\Box} t'$ *and* $a \in ext_T$ *then* $\exists s'. s \xrightarrow{a}_{\Box} s' \wedge (s', t') \in R$.
if $s \xrightarrow{a}_{\Diamond} s'$ *and* $a \in ext_S$ *then* $\exists t'. t \xrightarrow{\tau}^{*}_{\Diamond} t'. \exists t''. t' \xrightarrow{a}_{\Diamond} t'' \wedge (s', t'') \in R$.
if $s \xrightarrow{a}_{\Diamond} s'$ *and* $a \in int_S$ *then* $\exists t'. t \xrightarrow{\tau}^{*}_{\Diamond} t'. (s', t') \in R$

Observational modal refinement $\leq^{*}_{\mathrm{m}}$*is defined as the largest such relation. We say that a modal automaton S observationally refines a modal automaton T if there exists an observational modal refinement containing $(start_S, start_T)$.*

Interface Automata [1] can be considered a subset of modal automata in which the external actions ext_S are partitioned into inputs in_S and outputs out_S.

Definition 4 (Interface Automaton). *An* interface automaton *P is a tuple $P = (states_P, start_P, in_P, int_P, out_P, \rightarrow_P)$ where $states_P$ is a finite set of states, $start_P \in states_P$ is the initial state, in_P, out_P and int_P are three pairwise disjoint sets of input, output and hidden (internal) actions respectively, and $\rightarrow_P \subseteq states_P \times act_P \times states_P$ is the set of transitions where $act_P = in_P \cup out_P \cup int_P$.*

We require that the transition relation is input-deterministic such that for all $s, s', s'' \in states_P$ and all input actions $a \in in_P$ if $s \xrightarrow{a?} s'$ and $s \xrightarrow{a?} s''$ then $s' = s''$.

Similarly as for Modal Automata we define $s \xrightarrow{\tau}^{*} s'$ for Interface Automata to mean that there exists a sequence of internal transitions leading from s to s'. We define *alternating simulation* for interface automata as commonly used in software specification [9], which is slightly less general than the original [1]:

Definition 5 (Alternating Simulation). *For a pair of interface automata S and T with the same signature, a binary relation $R \subseteq states_S \times states_T$ is an alternating simulation if whenever sRt and $a \in act_S$ it holds that:*

if $t \xrightarrow{a?} t'$ *and* $a \in in_T$ *then* $\exists s'. s \xrightarrow{a?} s'$ *and* $(s', t') \in R$
if $s \xrightarrow{a!} s'$ *and* $a \in out_S$ *then* $\exists t'. t \xrightarrow{\tau}^{*} t'. \exists t''. t' \xrightarrow{a} t''$ *and* $(s, t'') \in R$

if $s \xrightarrow{a;} s'$ *and* $a \in int_S$ *then* $\exists t'.t \xrightarrow{\tau}{}^{*} t'$ *and* $(s', t') \in R$
Alternating simulation $\leq_{\mathrm{a}}$*is defined as the largest such relation. We say that* S *simulates* T*, written* $S \leq_{\mathrm{a}} T$*, if there exists an alternating simulation containing* $(start_S, start_T)$*.*

In order to compare interface automata with modal automata, we construct a translation function $\mathcal{T}$ mapping from the former to the latter. The result of the translation always fulfills the conditions listed below. It is easy to see that for modal automata that fulfill these conditions a reversed mapping can be constructed, too.

1. The may transition relation is input enabled, meaning that for each state $s \in states_S$ and each input action $a \in in_S$ there exists a state s' and a may transition $s \xrightarrow{a?}_{\Diamond} s'$
2. The constructed modal automaton is syntactically consistent: $\longrightarrow_{\Box} \subseteq \longrightarrow_{\Diamond}$
3. Must transitions are only labeled by inputs: $\longrightarrow_{\Box S} \subseteq states_S \times in_S \times states_S$

Let s_{mayall} be a fresh state that allows all behavior but does not require any behavior. If U denotes the universe of all inputs, such that for all interface automata P, $in_P \in U$, then we define the translation function as follows:

$$\mathcal{T}(states_P, start_P, in_P, out_P, int_P, \rightarrow_P) = (states_S, start_S, ext_S, int_S, \longrightarrow_{\Diamond}, \longrightarrow_{\Box})$$

where $states_S = states_P \cup \{s_{mayall}\}$, $start_S = start_P$, $ext_S = U \cup out_P$, $int_S = int_P$
and $s_1 \xrightarrow{a}{}^{S}_{\Diamond} s_2$ if $s_1 \xrightarrow{a}{}^{P} s_2$ and $a \in out_P \cup int_P$
and $s_3 \xrightarrow{a}{}^{S}_{\Box} s_4$ and $s_3 \xrightarrow{a}{}^{S}_{\Diamond} s_4$ if $s_3 \xrightarrow{a}{}^{P} s_4$ and $a \in in_P$
and $s_3 \xrightarrow{a}{}^{S}_{\Diamond} s_{mayall}$ if $\forall s' \in states_P (s_3, a, s') \notin \rightarrow^{P}$ and $a \in U$,
and s_{mayall} is a fresh state such that $\forall a \in act_S . s_{mayall} \xrightarrow{a}{}^{S}_{\Diamond} s_{mayall}$.

Theorem 6. *Alternating simulation and observational modal refinement coincide for interface automata in the following sense:*

$$\text{\textit{for any two interface automata} } S, T\text{: } S \leq_{\mathrm{a}} T \text{ \textit{iff} } \mathcal{T}(S) \leq^{*}_{\mathrm{m}} \mathcal{T}(T) \qquad (1)$$

Theorem 6 suggests that the usefulness of game theoretical models for component theories does not lie in its conformance relation. The crux is the use of control information in synthesis algorithms, when paths to error states are pruned. If this is the case we can construct an interface theory based on modal refinement and modal automata augmented with control information. Since modal refinement is richer and we can use a generalization of the synthesis algorithm used for interface automata, we will obtain a more expressive interface theory.

The fact that alternating simulation coincides with the *observational* version of modal refinement is expected, because Definition 5 embeds a closure on internal transitions. In fact in the absence of internal actions alternating simulation coincides with the regular modal refinement, as described in Definition 2, which is easy to prove. In order to simplify the developments we use the regular modal refinement ($\leq_{\mathrm{m}}$) from now on, even though most of our theorems can reasonably be considered for the observational refinement ($\leq^{*}_{\mathrm{m}}$), too.

4 Modal I/O Automata

Let us now define modal I/O automata, an extension of modal automata with control information, that will be the main ingredients of our interface theory and the product line theory coming in the next sections.

Definition 7. *A modal I/O automaton S is a tuple $S = (\mathit{states}_S, \mathit{start}_S, \mathit{in}_S, \mathit{out}_S, \mathit{int}_S, \longrightarrow_\Diamond, \longrightarrow_\Box)$, where states_S is a set of states, $\mathit{start}_S \in \mathit{states}_S$ is an initial state, in_S, out_S and int_S are pairwise disjoint sets of inputs, outputs and internal actions respectively ($\mathit{act}_S = \mathit{in}_S \cup \mathit{out}_S \cup \mathit{int}_S$), $\longrightarrow_{\Diamond S} \subseteq \mathit{states}_S \times \mathit{act}_S \times \mathit{states}_S$ is a may-transition relation, and $\longrightarrow_{\Box S} \subseteq \mathit{states}_S \times \mathit{act}_S \times \mathit{states}_S$ is a must-transition relation. Like previously we only consider syntactically consistent modal I/O automata here, so $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$.*

The composition for modal I/O automata combines both the modal aspects and the communications aspects. Two modal I/O automata S_1,S_2 are *composeable* iff their actions only overlap on complementary types: $(\mathit{in}_{S_1} \cup \mathit{int}_{S_1}) \cap (\mathit{in}_{S_2} \cup \mathit{int}_{S_2}) = \emptyset$ and $(\mathit{out}_{S_1} \cup \mathit{int}_{S_1}) \cap (\mathit{out}_{S_2} \cup \mathit{int}_{S_2}) = \emptyset$. The composition $S_1 \otimes S_2$ gives rise to a modal I/O automaton S such that $\mathit{states}_S = \mathit{states}_{S_1} \times \mathit{states}_{S_2}$, $\mathit{start}_S = (\mathit{start}_{S_1}, \mathit{start}_{S_2})$, $\mathit{in}_S = (\mathit{in}_{S_1} \setminus \mathit{out}_{S_2}) \cup (\mathit{in}_{S_2} \setminus \mathit{out}_{S_1})$, $\mathit{out}_S = (\mathit{out}_{S_1} \setminus \mathit{in}_{S_2}) \cup (\mathit{out}_{S_2} \setminus \mathit{in}_{S_1})$, $\mathit{int}_S = \mathit{int}_{S_1} \cup \mathit{int}_{S_2} \cup (\mathit{in}_{S_1} \cap \mathit{out}_{S_2}) \cup (\mathit{out}_{S_1} \cap \mathit{in}_{S_2})$. The transition relations are given by the following rules (see Fig. 2 for an example):

$$\frac{s_1 \xrightarrow{a!}_\gamma s_1' \qquad s_2 \xrightarrow{a?}_\gamma s_2'}{s_1 \otimes s_2 \xrightarrow{a}_\gamma s_1' \otimes s_2'}\ \gamma \in \{\Box, \Diamond\} \qquad\qquad \frac{s_1 \xrightarrow{a?}_\gamma s_1' \qquad s_2 \xrightarrow{a!}_\gamma s_2'}{s_1 \otimes s_2 \xrightarrow{a}_\gamma s_1' \otimes s_2'}\ \gamma \in \{\Box, \Diamond\}$$

$$\frac{s_1 \xrightarrow{a}_\gamma s_1' \qquad a \notin \mathit{act}_{S_2}}{s_1 \otimes s_2 \xrightarrow{a}_\gamma s_1' \otimes s_2}\ \gamma \in \{\Box, \Diamond\} \qquad\qquad \frac{s_2 \xrightarrow{a}_\gamma s_2' \qquad a \notin \mathit{act}_{S_1}}{s_1 \otimes s_2 \xrightarrow{a}_\gamma s_1 \otimes s_2'}\ \gamma \in \{\Box, \Diamond\}$$

For technical reasons (efficiency and simplicity) we always assume that unreachable states are removed after computing a composition (both here and in later sections). The following theorem is a simple corollary from the general fact that the modal refinement is a precongruence [15, 16]:

Theorem 8. *Modal refinement is a precongruence with respect to the above composition operator: for any four modal I/O automata T_1, T_2, S_1, S_2 such that $T_1 \leq_{\mathrm{m}} S_1$ and $T_2 \leq_{\mathrm{m}} S_2$ it holds that $T_1 \otimes T_2 \leq_{\mathrm{m}} S_1 \otimes S_2$.*

The composition operator ($\otimes$) defined above corresponds to a usual composition of software (hardware) *components*. Whenever we use it below we mean an unrestricted connection of components, which does not preclude deadlocks or other kinds of errors. We shall soon introduce two seemingly similar composition operators, ($|$) and ($\cdot$) having a very different use. In fact they are algorithms synthesizing *specifications* of how a result of simple composition ($\otimes$) should be used in order to guarantee the absence of certain errors.

5 A Modal Interface Theory

Interface theories support component based development. The aim is to specify component interfaces and from these interfaces to derive the interfaces of composite components. The novel aspect of the interface theory presented here is that the components can specify both required and allowed behavior, consequently it is suitable for expressing liveness properties.

In our specific interface theory an interface is given by a modal I/O automaton. A given interface specifies a set of potential implementations (concrete implementations have identical transition relations $\longrightarrow_{\Diamond} = \longrightarrow_{\Box}$). The goal of our interface theory is to be able to use interface descriptions to describe legal implementations of components in a component based system. The implementation relation, the relation that specifies which implementations conform to a given interface description is modal refinement $\leq_{\mathrm{m}}$. From the interface descriptions of two components it should be possible to derive the interface of the combined component. This is done without knowing more about the implementations, than the fact that they conform to their individual interface specification.

The result of composing two interfaces is a subset of the result of composing two modal I/O automata, in which all possible internally controllable paths leading to error states are removed. An *error state* is a state in which one component can output something that the other component might be unable to receive:

$$\begin{aligned} \mathit{err}^{\mathrm{i}}_{S_1,S_2} = \{(s_1,s_2) \in \mathit{states}_{S_1\otimes S_2} \,|\, &\text{there exists } a \in \mathit{int}_{S_1\otimes S_2} \text{ and states } s_1',\, s_2' \\ &\text{such that } (s_1 \xrightarrow{a!}{}^{S_1}_{\Diamond}\, s_1' \text{ and } s_2 \not\xrightarrow{a?}{}^{S_2}_{\Box}) \text{ or } (s_2 \xrightarrow{a!}{}^{S_2}_{\Diamond}\, s_2' \text{ and } s_1 \not\xrightarrow{a?}{}^{S_1}_{\Box})\} \end{aligned} \tag{2}$$

State 22 on Fig. 2 is an error state, witnessed by the fail action.

We are now ready to define the set of states of the composition:

$$\mathit{states}_{S_1|S_2} = \bigcap_{n=0}^{\infty} \mathit{prune}^{n}_{\mathrm{i}}(\mathit{states}_{S_1\otimes S_2} \setminus \mathit{err}^{\mathrm{i}}_{S_1,S_2}) \ , \tag{3}$$

where $\mathit{prune}_{\mathrm{i}}(S) = \{s \in S \,|\, \forall s'\, \forall a \in \mathit{int}_{S_1\otimes S_2}.\, s \xrightarrow{a}_{\Diamond} s' \text{ implies } s' \in S\}$, which is a monotonic function that removes, from the set of states S, all those states that in one internally controllable step may reach a state that is not in S.

See Figure 3 (left) for an example of how pruning works. State 22 has been removed as an error state, then state 21 was pruned as an error state can be reached from it by the internally controllable transition log!. Then all transitions involving states 21 and 22 were removed. State 20 remains in the result as the must transition labeled down is externally controllable.

Definition 9 (Composition). *The composition of two interfaces S_1 and S_2 is defined if S_1 and S_2 are composable modal I/O automata and $\mathit{start}_{S_1\otimes S_2} \in \mathit{states}_{S_1|S_2}$ (see above). The composition results in a modal I/O automaton $S_1|S_2$ such that $S_1|S_2 = (\mathit{states}_{S_1|S_2}, \mathit{start}_{S_1\otimes S_2}, \mathit{in}_{S_1\otimes S_2}, \mathit{out}_{S_1\otimes S_2}, \mathit{int}_{S_1\otimes S_2}, \longrightarrow^{S_1\otimes S_2}_{\Diamond} \cap (\mathit{states}_{S_1|S_2} \times \mathit{act}_{S_1\otimes S_2} \times \mathit{states}_{S_1|S_2}), \longrightarrow^{S_1\otimes S_2}_{\Box} \cap (\mathit{states}_{S_1|S_2} \times \mathit{act}_{S_1\otimes S_2} \times \mathit{states}_{S_1|S_2}))$.*

Two interfaces are compatible if the set of states resulting from composition, $\mathit{states}_{S_1|S_2}$, contains the initial state $(\mathit{start}_{S_1}, \mathit{start}_{S_2})$.

A desirable property of an interface theory is that components can be implemented independently of each other once the specifications are known. The following theorem formally states that this theory satisfies the property.

Theorem 10 (Independent Implementability). *For any two compatible interfaces S_1, S_2 and for any two implementations I_1, I_2, $I_1 \leq_{\mathrm{m}} S_1$ and $I_2 \leq_{\mathrm{m}} S_2$, it holds that $I_1 \otimes I_2 \leq_{\mathrm{m}} S_1|S_2$.*

This has three implications. First, $I_1 \otimes I_2$ would deliver all the required behavior promised by $S_1|S_2$ as long as it interacts with an environment obeying $S_1|S_2$. Second, $I_1 \otimes I_2$ will not do anything that $S_1|S_2$ would not allow in such an environment. Third, since $S_1|S_2$ does not contain error states then $I_1 \otimes I_2$ will not deadlock.

Theorem 11 (Deadlock Freeness Preservation). *For any two compatible interfaces S_1, S_2, any two implementations I_1, I_2, so $I_1 \leq_{\mathrm{m}} S_1$ and $I_2 \leq_{\mathrm{m}} S_2$, and any interface T compatible with $S_1|S_2$, if $T \otimes (S_1|S_2)$ has no reachable error states then $T \otimes (I_1 \otimes I_2)$ has no reachable error states.*

Finally the composition operator $(|)$ is commutative and associative up to graph isomorphism.

6 A Product Line Theory

In product line development one typically maintains a family of existing *assets* that are composed in a bottom-up fashion in order to build a product. Here we assume that existing assets are sufficient to build the product and no genuinely new programming is required. Assets are organized in small subfamilies, that can be thought of as configurable components. Choosing an asset from a subfamily is a configuration process. We model subfamilies as modal I/O automata, and call them *variability models*, to distinguish them from interfaces. The configuration process amounts to finding a suitable modal refinement of a variability model.

There is a need for a mechanism for composing variability models, to enable reasoning about the products that can be constructed using available assets. As in the interface theory we are interested in computing the legal uses for the composition of two models, without reaching error states. However we weaken the requirement this time: we do not require that *all* possible pairs of implementations give an error free composition, but only that there *exists* a pair of implementations that can avoid errors under a suitable use.

Two variability models are composable if their input, output and hidden actions do not overlap (the general rule for modal I/O automata). Two composable families can be composed, resulting in a description of a higher level component family. The signature of this variability model is found in the same way as for modal I/O automata. The requirement for the description of this more abstract family is that a specification that refines its description can be realized by choosing some concrete implementations from both lower level families involved. So that in effect one can configure the final product by configuring the abstract composed variability model, being sure that the selected configuration can be refined to configurations of each of the smaller components, available in the collection of assets. We give a sufficient condition for a refinement of a variability model to be decomposable.

The ultimate composition closely resembles the composition ($|$) for interface automata: it uses the regular modal I/O automata composition ($\otimes$) first and then removes error states. However now only internally controllable *required transitions* are pruned, while in the interface theory we had also removed states reachable by *allowed executions* of the same kind. The very existence of allowed internally controlled execution to an error state was considered dangerous in the interface theory—it is not in the product line theory. This is because we are not interested in eliminating errors by all means, but only in making sure that there exist error-free realizations of the specification. For two syntactically composable variability models we define the set of error states, $err^{\mathrm{v}}_{S_1,S_2}$, to be:

$$\begin{aligned} err^{\mathrm{v}}_{S_1,S_2} = \{(s_1,s_2) \in states_{S_1\otimes S_2} \mid & \text{ there exists } a \in int_{S_1\otimes S_2} \text{ and states } s_1', s_2' \\ & \text{such that } (s_1 \xrightarrow{a!}_{\Box} s_1' \text{ and } s_2 \not\xrightarrow{a?}_{\Diamond}) \text{ or } (s_1 \not\xrightarrow{a?}_{\Diamond} \text{ and } s_2 \xrightarrow{a!}_{\Box} s_2')\} \end{aligned} \quad (4)$$

In Figure 2 (right) state 22 is still an error state, though for a different reason than previously: in state 22 the *LinkLayer must* be able to produce fail, but the *Client* is *not allowed* to receive it. If a product of two variability models contains an error state it means that there exist configurations of composed assets that cannot safely work together. However, in the same spirit as in the interface theory, we can compute the set of legal uses that guarantee that there *exist* pairs of compatible configurations to interact with them. We remove from the product $S_1 \otimes S_2$ all the states that according to the variability specification *must* be able to reach an error state. If there is no states left then the two variability models are *incompatible*. Otherwise we arrive at a specification of states and transitions among the compatible states that constraint possible legal implementations obtained from these two families. Formally:

$$states_{S_1\cdot S_2} = \bigcap_{n=0}^{\infty} prune^n_{\mathrm{v}}(states_{S_1\otimes S_2} \setminus err^{\mathrm{v}}_{S_1,S_2}) \ , \quad (5)$$

where $prune_{\mathrm{v}}(S) = \{s \in S \mid \forall s'.\, \forall a \in int_{S_1\otimes S_2} \cup out_{S_1\otimes S_2}.\, s \xrightarrow{a}_{\Box} s' \text{ and } s' \in S\}$. We compute the two transition relations for the composition, by projecting the transition relations of the parallel composition $S_1 \otimes S_2$ onto the new set of states:

$$\longrightarrow^{S_1\cdot S_2}_{\Diamond} = \longrightarrow^{S_1\otimes S_2}_{\Diamond} \cap (states_{S_1\cdot S_2} \times act_{S_1\otimes S_2} \times states_{S_1\cdot S_2}) \quad (6)$$

$$\longrightarrow^{S_1\cdot S_2}_{\Box} = \longrightarrow^{S_1\otimes S_2}_{\Box} \cap (states_{S_1\cdot S_2} \times act_{S_1\otimes S_2} \times states_{S_1\cdot S_2}) \ . \quad (7)$$

Finally we can state the complete result of the composition: a modal I/O automaton $S_1 \cdot S_2$ such that $S_1 \cdot S_2 = (\mathit{states}_{S_1 \cdot S_2}, (\mathit{start}_{S_1}, \mathit{start}_{S_2}), \mathit{in}_{S_1 \otimes S_2}, \mathit{out}_{S_1 \otimes S_2}, \mathit{int}_{S_1 \otimes S_2}, \longrightarrow_{\Diamond}^{S_1 \cdot S_2}, \longrightarrow_{\Box}^{S_1 \cdot S_2})$ and all the components are defined above.

Definition 12. *Two variability models are compatible if they are composable and their composition is nonempty.*

It turns out that *observationally consistent* refinements of compositions of variability models are realizable with existing assets. We define observational consistency for states of a single automaton. Let $t \xrightarrow{A}_{\Box}{}^{*} t'$ mean that t' is reachable from t via a possible empty sequence of required transitions labeled by possibly different actions from a set A.

Definition 13. *Let T be a modal automaton and let $A \subseteq \mathit{act}_T$ be a set of actions. A relation $C \subseteq \mathit{states}_T \times \mathit{states}_T$ is an observational consistency relation with respect to A if for any pair of states $(t_1, t_2) \in C$ the following two properties hold:*

1. $\forall t_1'.$ *if* $t_1 \xrightarrow{A}{}_{\Box}^{*} t_1'$ *then* $\forall a \notin A. \forall t_1''. t_1' \xrightarrow{a}_{\Box} t_1''$ *implies* $\exists t_2'. t_2 \xrightarrow{a}_{\Diamond} t_2' \wedge (t_1'', t_2') \in C.$
2. $\forall t_2'.$ *if* $t_2 \xrightarrow{A}{}_{\Box}^{*} t_2'$ *then* $\forall a \notin A. \forall t_2''. t_2' \xrightarrow{a}_{\Box} t_2''$ *implies* $\exists t_1'. t_1 \xrightarrow{a}_{\Diamond} t_1' \wedge (t_1', t_2'') \in C.$

Two states are observationally consistent if there exists an observational consistency relation relating them. A set of states is said to be observationally consistent with respect to A if all possible pairs of states from the set are observationally consistent with respect to A. An automaton T is observationally consistent with respect to A iff the set $\{\mathit{start}_T\}$ is an observationally consistent set.

The following theorem states the existence of decomposition formally:

Theorem 14 (Decomposability). *Let T_1, T_2 be deterministic composable variability models, and S be a configuration (a deterministic variability model itself) such that $S \leq_{\mathrm{m}} T_1 \cdot T_2$, and T_1, S are observationally consistent with respect to $\mathit{act}_{T_1} \setminus \mathit{act}_{T_2}$ and T_2, S are observationally consistent with respect to $\mathit{act}_{T_2} \setminus \mathit{act}_{T_1}$. Then there exist S_1 and S_2 such that $S_1 \leq_{\mathrm{m}} T_1$ and $S_2 \leq_{\mathrm{m}} T_2$ and $S_1 \otimes S_2 \leq_{\mathrm{m}} S$.*

A version of the theorem, not requiring observational consistency, does not hold, which can be demonstrated with a counter-example, not included here.

An important corollary is that the decomposition can be carried over down to precise configurations: if a concrete configuration of a product is required, then there exist concrete configurations of assets to realize it. The question whether a specification is realizable with given assets is reduced to establishing observational consistency and a modal refinement between the postulated requirement and the variability model. Consequently the abstract variability model can be communicated to configuration engineers and used to configure final products.

Let us close our discussion with a statement that the $(\cdot)$ operator is general enough to describe all implementations safely realizable with existing assets.

Theorem 15 (Completeness). *For any two compatible variability models T_1, T_2 and any two compatible concrete implementation specifications I_1, I_2, where $I_1 \leq_{\mathrm{m}} T_1$ and $I_2 \leq_{\mathrm{m}} T_2$ it holds that $I_1 \cdot I_2 \leq_{\mathrm{m}} T_1 \cdot T_2$.*

7 Conclusion & Future Work

We have investigated the relation between alternating simulation as used in interface automata and observational modal refinement, concluding that former is a case of the latter. We have argued that the strength of the game theoretic approach to interface theories does not lie in alternating refinement itself, but in the labeling of transitions with control information; in partitioning the actions into internally and externally controllable. We have extended modal transition systems with this information and demonstrated that in this way interface theories tracking liveness properties, can be built. Finally we have presented a product line theory describing variability in behavior of component families.

In the future we would like to extend the product line theory of Section 6 to a full featured theory based on observational modal refinement and study its properties in depth. Also it appears interesting to investigate the relation between the general notion of alternating refinement [8] and (modal) transition systems, lifting the restrictions accepted in Section 3 after the interface automata model.

References

1. Alfaro, L., Henzinger, T.A.: Interface automata. In: Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE), Vienna, Austria, ACM Press (2001) 109–120
2. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, IEEE Computer Society (1988)
3. Chakabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.I.A.: Resource interfaces. In Alur, R., Lee, I., eds.: EMSOFT 03: 3rd Intl. Workshop on Embedded Software. LNCS, Springer (2003)
4. Alfaro, L., Henzinger, T., Stoelinga, M.I.A.: Timed interfaces. In Sangiovanni-Vincentelli, A., Sifakis, J., eds.: EMSOFT 02: 2nd Intl. Workshop on Embedded Software. LNCS, Springer (2002)
5. Larsen, K.G., Nyman, U., Wąsowski, A.: Interface input/output automata. In Misra, J., Nipkow, T., Sekerinski, E., eds.: 14th International Symposium on Formal Methods (FM) Hamilton, Canada, August 21–27, 2006 Proceedings. Volume 4085 of LNCS., Springer (2006) 82–97
6. Černá, I., Vařeková, P., Zimmerová, B.: Component substitutability via equivalencies of component-interaction automata. In: FACS'06. (2006) 115–130 To be published in ENTCS.
7. Hermanns, H., Rehof, J., Stoelinga, M.I.A., eds.: Workshop Procedings FIT 2005: Foundations of Interface Technologies. ENTCS, Elsevier Science Publishers (2005)
8. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.: Alternating refinement relations. In Sangiorgi, D., de Simone, R., eds.: Proceedings of the Ninth International Conference on Concurrency Theory (CONCUR'98). Volume 1466 of LNCS., Springer (1998) 163–178
9. Alfaro, L., Henzinger, T.A.: Interface-based design. In: In Engineering Theories of Software Intensive Systems, Marktoberdorf Summer School, Kluwer Academic Publishers (2004)
10. Carrez, C., Fantechi, A., Najm, E.: Assembling components with behavioral contracts. Annales del Télécommunications **60** (2005)
11. Parnas, D.L.: On the design and development of program families. IEEE Transactions on Software Engineering **Vol. SE-2** (1976) 1–9
12. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley (2000)
13. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering—Foundations, Principles, and Techniques. Springer (2005)
14. Larsen, K.G., Larsen, U., Wąsowski, A.: Color-blind specifications for transformations of reactive synchronous programs. In Cerioli, M., ed.: FASE, Edinburgh, April 2005. LNCS, Springer (2005)
15. Boudol, G., Larsen, K.G.: Graphical versus logical specifications. In Arnold, A., ed.: CAAP. Volume 431 of Lecture Notes in Computer Science., Springer (1990) 57–71
16. Larsen, K.G.: Modal specifications. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science., Springer (1989) 232–246

A Proofs

This appendix contains proofs of theorems and lemmas, along with some counterexamples for negative claims or one-way implications. *The appendix is not an integral part of the paper, and reading it is not required in order to assess the value of the results.*

A.1 Appendix for Section 3

This section uses formulations of Alternating Simulation and Observational Modal Refinement with ϵ-*closure*(s) instead of $s\xrightarrow{\tau}^{*}$.

Proof (of Theorem 6). The proof will be divided into two directions. First we will prove that

$$\forall S, T \in \text{IA}.\, S \leq_{\text{a}} T \implies \mathcal{T}(S) \leq^{*}_{\text{m}} \mathcal{T}(T).$$

We will prove this by showing that alternating simulation is a subset of observational modal refinement on the translation of IA: $\leq_{\text{a}} \subseteq \leq^{*}_{\text{m}}$. This will be shown by showing that the following relation is a modal refinement.

$$R = \{(s,t) | \exists \hat{s}, \hat{t}. s = \mathcal{T}(\hat{s}) \wedge t = \mathcal{T}(\hat{t}) \wedge \hat{s} \leq_{\text{a}} \hat{t}\} \cup \{(s, s_{mayall}) | s \in states_S\}$$

This is shown in three different cases, one for each of the rules that define observational modal refinement.

1. **Must transition, external action:** Take $t.t\xrightarrow{a}_{\square}t' \wedge a \in ext_T$. We can conclude from the definition of translation that this case only exists for $a \in in_T$. By R we have that $\exists \hat{t}.\hat{t}\xrightarrow{a?}\hat{t}'$. From the definition of Alternating Simulation we have that $\exists \hat{s}.\hat{s}\xrightarrow{a?}\hat{s}' \wedge (s', t') \in R$. By translation we have that $s\xrightarrow{a}_{\square}s'$ and this implies that $(s', t') \in R$.
2. **May transition, external action:** Take $s.s\xrightarrow{a}_{\diamond}s' \wedge a \in out_S \cup in_S$ it means, by R, that $\exists \hat{s}.\hat{s}\xrightarrow{a}\hat{s}'$

   2.1 $a \in out_S \wedge \hat{s}\xrightarrow{a!}\hat{s}'$, by $\hat{s} \leq_{\text{a}} \hat{t}$ and the definition of alternating simulation we have that $\hat{t}\xrightarrow{a!}\hat{t}' \wedge \hat{s}' \leq_{\text{a}} \hat{t}'$. By translation we have $t\xrightarrow{a}_{\diamond}t'$ this all implies that $(s', t') \in R$.

   2.2 $a \in in_S \wedge \hat{s}\xrightarrow{a?}\hat{s}' \wedge \hat{t}\xrightarrow{a?}\hat{t}'$, by $\hat{s} \leq_{\text{a}} \hat{t}$, the definition of alternating simulation and the fact that IA are input deterministic we have that $\hat{s}\xrightarrow{a?}\hat{s}' \wedge \hat{s}' \leq_{\text{a}} \hat{t}'$ and this implies that $(s', t') \in R$.

   2.3 $a \in in_S \wedge \hat{s}\xrightarrow{a?}\hat{s}' \wedge \hat{t} \not\xrightarrow{a?}$, by translation we have $t\xrightarrow{a?}_{\diamond}s_{mayall}$ and by definition of R we have that $(s, s_{mayall}) \in R$
3. **May transition, internal action:** Take $s.s\xrightarrow{a}_{\diamond}s' \wedge a \in int_S$ it means, by R and translation, that $\hat{s}\xrightarrow{a;}\hat{s}' \wedge s = \mathcal{T}(\hat{s})$. By the definition of alternating simulation we have that $\exists \hat{t}'.\hat{t}\xrightarrow{\tau}^{*}\hat{t}' \wedge \hat{s}' \leq_{\text{a}} \hat{t}'$. By translation we have that $\exists t'.t\xrightarrow{\tau}^{*}_{\diamond}t'.t' = \mathcal{T}(\hat{t}')$. This all implies $(s', t') \in R$.

We will now prove the other direction:

$$\forall S, T \in \text{IA}.\; S \leq_{\text{a}} T \iff \mathcal{T}(S) \leq^*_{\text{m}} \mathcal{T}(T).$$

We will prove this by showing that observational modal refinement, on the translation of IA, is a subset of alternating : $\leq^*_{\text{m}} \subseteq \leq_{\text{a}}$. This will be shown by showing that the following relation is an alternating simulation.

$$Q = \{(\hat{s}, \hat{t}) | \exists s, t. s = \mathcal{T}(\hat{s}) \land t = \mathcal{T}(\hat{t}) \land s \leq^*_{\text{m}} t\}$$

This will be split into three cases, one for each of the rules in the definition of Alternating Simulation.

1. Take $\hat{t} \xrightarrow{a?} \hat{t}'$ by Q and translation we have that $a \in in_T \land t \xrightarrow{a}_{\Box} t'$. We have by $s \leq^*_{\text{m}} t$ and the definition of Observational Modal Refinement that $\exists s'. s \xrightarrow{a}_{\Box} s' \land s' \leq_{\text{m}} t'$ and by translation we have that $\hat{s} \xrightarrow{a} \hat{s}'$ where $s' = \mathcal{T}(\hat{s}')$ which implies that $(\hat{s}', \hat{t}') \in Q$.
2. Take $\hat{s} \xrightarrow{a!} \hat{s}'$, by Q and translation we have that $a \in out_S . s \xrightarrow{a}_{\Diamond} s'$. We have by $s \leq^*_{\text{m}} t$ and the definition of Observational Modal Refinement that $\exists t'. t \xrightarrow{\tau}^*_{\Diamond} t'$. $\exists t''. t' \xrightarrow{a}_{\Diamond} t''$ and $s' \leq^*_{\text{m}} t''$. By translation we have that this will give rise to a sequence of internal transitions followed by an a transition such that we know that $\exists \hat{t}'. \hat{t} \xrightarrow{\tau}{}^* \hat{t}'. \hat{t}' \xrightarrow{a!} \hat{t}'' \land \hat{s}' \leq_{\text{a}} \hat{t}''$ This all implies that $(\hat{s}', \hat{t}'') \in Q$.
3. Take $\hat{s} \xrightarrow{a;} \hat{s}'$. By Q and by translation we have that $a \in int_S \land s \xrightarrow{a}_{\Diamond} s'$ We have by $s \leq^*_{\text{m}} t$ and the definition of Observational Modal Refinement that $\exists t'. t \xrightarrow{\tau}^*_{\Diamond} t' \land s' \leq^*_{\text{m}} t'$. By translation we know that this sequence of zero or more internal transitions will give rise to an identical sequence of internal transitions such that $\exists \hat{t}'. \hat{t} \xrightarrow{\tau}{}^* \hat{t}'$ and $\hat{s}' \leq_{\text{a}} \hat{t}'$. This all implies that $(\hat{s}', \hat{t}') \in Q$

□

A.2 Appendix for Section 4

Lemma 16. *For any two composeable and syntactically consistent modal I/O automata S_1, S_2 their parallel composition $S_1 \otimes S_2$ is also syntactically consistent.*

A.3 Appendix for Section 5

Proof (of Theorem 10). This theorem is proven by showing that the relation R is a modal refinement:

$$R = \{(i, s) \in states_{I_1 \otimes I_2} \times states_{S_1 | S_2} | i = (i_1, i_2) \land s = (s_1, s_2) \land i_1 \leq_{\text{m}} s_1 \land i_2 \leq_{\text{m}} s_2\}$$

The proof is divided into two cases, one for each of the rules in the definition of modal refinement.

1. $s \xrightarrow{a}_{\Box} s'$. This means that $(s_1, s_2) \xrightarrow{a}_{\Box} (s'_1, s'_2)$.
   We want to show that $\exists i'.i \xrightarrow{a}_{\Box} i' \wedge (i', s') \in R$. This will be divided into five sub cases depending on how $(s_1, s_2) \xrightarrow{a}_{\Box} (s'_1, s'_2)$ is achieved. Several of these cases are symmetric versions of each other.
   1.1 $s_1 \xrightarrow{a!}_{\Box} s'_1 \wedge a \in \mathit{int}_{S_1|S_2}$. We know that $s_2 \xrightarrow{a?}_{\Box} s'_2$ must exists, else the output transition would have been pruned. We know $i_1 \leq_{\mathrm{m}} s_1 \wedge i_2 \leq_{\mathrm{m}} s_2$ which gives us $i_1 \xrightarrow{a!}_{\Box} i'_1 \wedge i_2 \xrightarrow{a?}_{\Box} i'_2$. So take $i = (i'_1, i'_2)$, by definition of $I_1 \otimes I_2$ we have that $i \xrightarrow{a}_{\Box} i'$ and this implies that $(i', s') \in R$.
   1.2 This case is completely symmetric, where it is s_2 that outputs.
   1.3 $s_1 \xrightarrow{a!}_{\Box} s'_1 \wedge a \in \mathit{out}_S \wedge a \in \mathit{ext}_{S_1|S_2}$ by $i_1 \leq_{\mathrm{m}} s_1$ we have that $i_1 \xrightarrow{a!}_{\Box} i'_1 \wedge i'_1 \leq_{\mathrm{m}} s'_1$. Also in this case we have, by composability, that $s'_2 = s_2 \wedge i'_2 = i_2$ and $(i_1, i_2) \xrightarrow{a!}_{\Box} (i'_1, i_2)$. For $i' = (i'_1, i_2) \wedge s' = (s'_1, s_2)$ this all implies that $(i', s') \in R$.
   1.4 $s_1 \xrightarrow{a?}_{\Box} s'_1 \wedge a \in \mathit{in}_S \wedge a \in \mathit{ext}_{S_1|S_2}$. This case is symmetric with the previous case.
   1.5 $s_1 \xrightarrow{a;}_{\Box} s'_1 \wedge a \in \mathit{int}_S \wedge a \in \mathit{int}_{S_1|S_2}$. This case is symmetric with the previous case. All three cases also have symmetric cases where the transition in question is part of S_2.
2. $i \xrightarrow{a}_{\Diamond} i'$ this means that $(i_1, i_2) \xrightarrow{a}_{\Diamond} (i'_1, i'_2)$.
   We want to show that $\exists s'.s \xrightarrow{a}_{\Diamond} s' \wedge (i', s') \in R$. This will be divided into five sub cases depending on how $(i_1, i_2) \xrightarrow{a}_{\Diamond} (i'_1, i'_2)$ is achieved. Several of these cases are symmetric versions of each other.
   2.1 $i_1 \xrightarrow{a!}_{\Diamond} i'_1 \wedge i_2 \xrightarrow{a?}_{\Diamond} i'_2$. By R and the definition of $\leq_{\mathrm{m}}$we have that $s_1 \xrightarrow{a!}_{\Diamond} s'_1 \wedge s_2 \xrightarrow{a!}_{\Diamond} s'_2 \wedge i'_1 \leq_{\mathrm{m}} s'_1 \wedge i'_2 \leq_{\mathrm{m}} s'_2$ which gives us that $((i'_1, i'_2), (s'_1, s'_2)) \in R$.
   2.2 This case is completely symmetric, where it is i_2 that outputs.
   2.3 $i_1 \xrightarrow{a!}_{\Diamond} i'_1 \wedge a \in \mathit{out}_I \wedge a \in \mathit{ext}_{I_1 \otimes I_2}$ by $i_1 \leq_{\mathrm{m}} s_1$ we have that $s_1 \xrightarrow{a!}_{\Diamond} s'_1 \wedge i'_1 \leq_{\mathrm{m}} s'_1$. Also in this case we have, by composability, that $s'_2 = s_2 \wedge i'_2 = i_2$ and $(s_1, s_2) \xrightarrow{a!}_{\Diamond} (s'_1, s_2)$. For $i' = (i'_1, i_2) \wedge s' = (s'_1, s_2)$ this all implies that $(i', s') \in R$.
   2.4 $i_1 \xrightarrow{a?}_{\Diamond} i'_1 \wedge a \in \mathit{in}_I \wedge a \in \mathit{ext}_{I_1 \otimes I_2}$. This case is symmetric with the previous case.
   2.5 $i_1 \xrightarrow{a;}_{\Diamond} i'_1 \wedge a \in \mathit{int}_I \wedge a \in \mathit{int}_{I_1 \otimes I_2}$. This case is symmetric with the previous case. All three cases also have symmetric cases where the transition in question is part of I_2.

□

Proof (of Theorem 11).

The proof proceeds as a contrapositive proof in which we show that if an error state was reachable in $T \otimes (I_1 \otimes I_2)$ then an error state would also be reachable in $T \otimes (S_1|S_2)$. There are two ways in which an error state could be reachable in $T \otimes (I_1 \otimes I_2)$.

1. $\mathit{err}^{\mathrm{i}}_{T,(I_1 \otimes I_2)} \cap \mathit{reachable}(T \otimes (I_1 \otimes I_2))$is non empty.

2. $\mathit{\Pi}_2(\mathit{reachable}(T \otimes (I_1 \otimes I_2))) \cap \mathit{err}^{\mathrm{i}}_{I_1,I_2}$ is non empty.

   Contrapositive proof:

1. Assume that $(t, i) \in \mathit{err}^{\mathrm{i}}_{T,(I_1 \otimes I_2)}$ and that (t, i) is reachable. No we want to show that $\exists(t, s) \in \mathit{err}^{\mathrm{i}}_{T,(S_1|S_2)}$ and that (t, s) is reachable.
   Because t is reachable and $I_1 \otimes I_2 \leq_{\mathrm{m}} S_1|S_2$ (Theorem 10) we know that $\exists s \in \mathit{states}_{S_1|S_2}$ and $i \leq_{\mathrm{m}} s \wedge s$ is reachable by *may* transitions in $S_1|S_2$.
   1.1 $t \xrightarrow{a!}_{\Diamond} t' \wedge i \not\xrightarrow{a?}_{\Box} \wedge a \in \mathit{int}_{T \otimes (I_1 \otimes I_2)}$ but then $s \not\xrightarrow{a?}_{\Box}$. We now need to argue that (t, s) is reachable by *may* transitions. This follows from $I_1 \otimes I_2 \leq_{\mathrm{m}} S_1|S_2$ (Theorem 10). Because of consistency we only consider *may* transitions.
   Executions of T and $I_1 \otimes I_2$ is a sequence of may transitions of T and $I_1 \otimes I_2$. All the *may* transitions of $I_1 \otimes I_2$ can be matched by *may* transitions of $S_1|S_2$
   1.2 $i \xrightarrow{a!}_{\Diamond} i' \wedge t \not\xrightarrow{a?}_{\Box} \wedge a \in \mathit{int}_{T \otimes (I_1 \otimes I_2)}$. The argument is identical to the previous case.
2. Assume that $i_1 \xrightarrow{a!}_{\Diamond} i_1' \wedge i_2 \not\xrightarrow{a?}_{\Box}$ and $\exists t.(t, i_1, i_2)$ is reachable. This implies that $s_1 \xrightarrow{a!}_{\Diamond} s_1' \wedge s_2 \not\xrightarrow{a?}_{\Box}$. So we can conclude that an error state would be reachable in $T \otimes (S_1|S_2)$ in this case.

Lemma 17. *For any two composeable and syntactically consistent modal interface automata S_1, S_2 their parallel composition $S_1|S_2$ is also syntactically consistent.*

Theorem 18 (Associativity). $\forall S_1, S_2, S_3$*. pairwise compatible* $S_1|(S_2|S_3)$ *is isomorphic with* $(S_1|S_2)|S_3$.

A.4 Appendix for Section 6

Lemma 19. *For any two composeable and syntactically consistent modal variability models S_1, S_2 their parallel composition $S_1 \cdot S_2$ is also syntactically consistent.*

Definition 20 (A-closure). *For a set of actions A we define an A-closure of a pair of states $(s, t_1) \in \mathit{states}_S \times \mathit{states}_{T_1}$ as a subset Σ of $\mathit{states}_S \times \mathit{states}_{T_1}$ consisting of (s, t_1) itself and all pairs (s', t_1') in which s' can be reached from s by following a sequence of steps from $\longrightarrow^{S}_{\Box}$ labeled solely by actions in A and t_1' can be reached from t_1 by following an identical sequence (sequence with the same labels) of steps from $\longrightarrow^{T_1}_{\Box}$. Closures for pairs of states of S and T_2 are defined analogously.*

Definition 21 (A-closure). *We lift definition 20 to sets of pairs of states, such that the result is simply the union of the A-closures of all pairs.*

Let $t\xrightarrow{A}{}_{\Box}{}^{*}t'$ mean that t' is reachable from t via a possible empty sequence of required transitions labeled by actions from a set A (possibly different actions).

We will define observational consistency for states of a single automata.

Definition 22. *Let T be a modal automaton and let $A \subseteq act_T$ be a set of actions. A relation $C \subseteq states_T \times states_T$ is an observational consistency relation with respect to A if for any pair of states $(t_1, t_2) \in C$ the following two properties hold:*

1. $\forall t'_1.\ \mathit{if}\ t_1\xrightarrow{A}{}_{\Box}^{*}t'_1\ \mathit{then}\ \forall a \notin A.\ \forall t''_1.\ t'_1\xrightarrow{a}{}_{\Box}t''_1\ \mathit{implies}\ \exists t'_2.\ t_2\xrightarrow{a}{}_{\Diamond}t'_2 \wedge (t''_1, t'_2) \in C.$
2. $\forall t'_2.\ \mathit{if}\ t_2\xrightarrow{A}{}_{\Box}^{*}t'_2\ \mathit{then}\ \forall a \notin A.\ \forall t''_2.\ t'_2\xrightarrow{a}{}_{\Box}t''_2\ \mathit{implies}\ \exists t'_1.\ t_1\xrightarrow{a}{}_{\Diamond}t'_1 \wedge (t'_1, t''_2) \in C.$

Two states are observationally consistent if there exists an observational consistency relation relating them. A set of states is said to be observationally consistent with respect to A if all possible pairs of states from the set are observationally consistent with respect to A.

An automaton T is observationally consistent with respect to A iff the set $\{start_T\}$ is an observationally consistent set.

Lemma 23. *Consistency is transitive in the following sense: for a consistency relation C if $(t_1, t_2) \in C$ and $(t_2, t_3) \in C$ then $(t_1, t_3) \in C$.*

Lemma 24. *Let S, T_1, T_2 be modal I/O automata and $S \leq_{\mathrm{m}} T_1 \cdot T_2$. If $s \in states_S$ and $t_2 \in states_{T_2}$ are observationally consistent states wrt to $act_{T_2} \setminus act_{T_1}$ then projections of $(act_{T_2} \setminus act_{T_1})$–closure$(s, t_2)$ on the first and second*[4] *component give observationally consistent sets of states with respect to the same set of actions $act_{T_2} \setminus act_{T_1}$.*

Similarly if $s \in states_S$ and $t_1 \in states_{T_1}$ are observationally consistent states wrt to $act_{T_1} \setminus act_{T_2}$ then projections of $(act_{T_1} \setminus act_{T_2})$–closure$(s, t_1)$ on the first and second component give observationally consistent sets of states with respect to the same set of actions $act_{T_1} \setminus act_{T_2}$.

These claims generalize also to sets of consistent states.

Proof (of Thm. 14). We shall construct S_1 and S_2 exhibiting the requirements of the theorem. The signatures of S_1 and S_2 are identical to those of T_1 and T_2:

$$int_{S_i} = int_{T_i}, \quad out_{S_i} = out_{T_i}, \quad int_{S_i} = int_{T_i} \ . \tag{8}$$

Since $S \leq_{\mathrm{m}} T_1 \cdot T_2$ there exists the least relation $R \subseteq states_S \times (states_{T_1} \times states_{T_2})$, which is a modal refinement of $T_1 \cdot T_2$ by S. Let

$$states_{S_1} = \{(\Sigma_1, t_1) \mid t_1 \in states_{T_1} \text{ and } \Sigma_1 \subseteq \{(s, t_2) \mid (s, (t_1, t_2)) \in R\}\} \tag{9}$$

$$states_{S_2} = \{(\Sigma_2, t_2) \mid t_2 \in states_{T_2} \text{ and } \Sigma_2 \subseteq \{(s, t_1) \mid (s, (t_1, t_2)) \in R\}\} \tag{10}$$

[4] For the current version of the proof we only need to claim consistency when projected on the first component.

and

$$\mathit{start}_{S_1} = (\Sigma_1^0, \mathit{start}_{T_1}), \text{ where } \Sigma_1^0 = (\mathit{act}_{T_2} \setminus \mathit{act}_{T_1})\text{-}\mathit{closure}(\mathit{start}_S, \mathit{start}_{T_2}) \tag{11}$$

$$\mathit{start}_{S_2} = (\Sigma_2^0, \mathit{start}_{T_2}), \text{ where } \Sigma_2^0 = (\mathit{act}_{T_1} \setminus \mathit{act}_{T_2})\text{-}\mathit{closure}(\mathit{start}_S, \mathit{start}_{T_1}) \tag{12}$$

We create only one transition relation for each of S_1 and S_2 (or more precisely both will have two, but identical transition relations). Intuitively this transition relation for S_1 will contain all steps allowed by T_1 and required by S. Formally it is given by the following rules:

$$\frac{\begin{array}{c} a \in \mathit{act}_{S_1} \setminus \mathit{act}_{S_2} \qquad t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t_1' \qquad \exists (s, t_2) \in \Sigma_1 .\, s \xrightarrow{a}{}^{S}_{\Box} \\ \Sigma_1' = \{(s', t_2) \mid \exists (s, t_2) \in \Sigma_1 .\, s \xrightarrow{a}{}^{S}_{\Diamond} s'\} \end{array}}{(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} ((\mathit{act}_{T_2} \setminus \mathit{act}_{T_1})\text{-}\mathit{closure}(\Sigma_1'), t_1')} \tag{13}$$

$$\frac{\begin{array}{c} a \in \mathit{act}_{S_1} \cap \mathit{act}_{S_2} \qquad t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t_1' \qquad \exists (s, t_2) \in \Sigma_1 .\, s \xrightarrow{a}{}^{S}_{\Box} \\ \Sigma_1' = \{(s', t_2') \mid \exists (s, t_2) \in \Sigma_1 .\, s \xrightarrow{a}{}^{S}_{\Diamond} s' \wedge t_2 \xrightarrow{a}{}^{T_2}_{\Diamond} t_2'\} \end{array}}{(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} ((\mathit{act}_{T_2} \setminus \mathit{act}_{T_1})\text{-}\mathit{closure}(\Sigma_1'), t_1')} \tag{14}$$

$$\frac{a \in \mathit{act}_{S_1} \cap \mathit{act}_{S_2} \qquad t_1 \xrightarrow{a}{}^{T_1}_{\Box} t_1' \qquad \forall (s, t_2) \in \Sigma_1 .\, s \not\xrightarrow{a}{}^{S}_{\Box}}{(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\emptyset, t_1')} \tag{15}$$

$$\frac{a \in \mathit{act}_{S_1} \setminus \mathit{act}_{S_2} \qquad t_1 \xrightarrow{a}{}^{T_1}_{\Box} t_1'}{(\emptyset, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\emptyset, t_1')} \tag{16}$$

We take the must transition relation $\longrightarrow^{S_1}_{\Box}$ to be identical with $\longrightarrow^{S_1}_{\Diamond}$. Note that effectively S_1 follows all must transition relations of S in its sort, except that whenever T_1 requires an input that is not followed by S (as T_2 is not able to synchronize on this input), we redirect the transition relation to a region where all must transitions of T_1 are mapped. We do that as minimum addition to maintain refinement of T_1 by S_1, on the functionality not explored by S.

We refrain from showing the rules for S_2 here—they can be easily constructed by analogy, as the problem is entirely symmetric.

It is clear that the constructed systems S_1 and S_2 are deterministic—the closure operation is deterministic and we apply to a unique maximal set for each action in each particular source state.

Lemma 25. *The rules for transitions of S_1 ensures that if the originating state belongs to states_{S_1} then the target state will also belong to states_{S_1}.*

An entirely symmetric lemma can be made for S_2.

Proof. (Lemma 25) First we need to argue that the initial state $\mathit{start}_{S_1} \in \mathit{states}_{S_1}$. Firstly $\mathit{start}_{T_1} \in \mathit{states}_{T_1}$ which satisfies the first part of the requirement for states in states_{S_1}. Now we need to show that $(\mathit{act}_{T_2} \setminus \mathit{act}_{T_1})\text{-}\mathit{closure}(\mathit{start}_S, \mathit{start}_{T_2}) \subseteq \{(s, t_2) \mid (s, (t_1, t_2)) \in R\}$. The state from which the closure is calculated namely, $(\{(\mathit{start}_S, \mathit{start}_{T_2})\}, \mathit{start}_{T_1})$, is part of states_{S_1} because $(\mathit{start}_{S_1}, (\mathit{start}_{T_1}, \mathit{start}_{T_2})) \in R$. All the transitions that are taken in the calculation of the closure are on actions not involving T_1 and are taken simultaneously by S and T_2, which ensures that all pairs of states Σ'_1 that are reached will still fulfill the requirement for being in states_{S_1}.

The rest of the proof consists of four cases, one for each rule. We need to argue for transitions generated by each of the four rules that the target state will be in states_{S_1}, given that the source state is. Transitions generated by rule (13) ensure this because the states that are in Σ'_1 have taken one transition that is on a non shared action of T_1. This transition is taken simultaneously by T_1 and S. Finally the closure also preserves the property, by the same argument as before. The argument for rule (14) is similar, the only difference being that the first transition is on a shared action and is taken by S, T_1 and T_2. Rule (15) and (16) are different. Here the argument is that $\emptyset$ is a subset of $\{(s, t_2) \mid (s, (t_1, t_2)) \in R\}$.

We want to show that 1° $S_1 \leq_{\mathrm{m}} T_1$, 2° $S_2 \leq_{\mathrm{m}} T_2$ and 3° $S_1 \otimes S_2 \leq_{\mathrm{m}} S$.

1° Show that

$$R_1 = \{((\Sigma_1, t_1), t_1) \,|\, \Sigma_1 \in \mathit{states}_{S_1} \text{ and } t_1 \in \mathit{states}_{T_1}\} \tag{17}$$

is a modal refinement of T_1 by S_1.
Consider an arbitrary pair of states $((\Sigma_1, t_1), t_1) \in R_1$ and a transition $t_1 \xrightarrow{a}{}^{T_1}_{\square} t'_1$. We want to show that there exists a state (Σ'_1, t'_1) and a transition such that $(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\square} (\Sigma'_1, t'_1)$ and $((\Sigma'_1, t'_1), t'_1) \in R_1$

1.1° If $\Sigma_1 = \emptyset$ then take Σ'_1 to be $\emptyset$ and the corresponding transition exists due to rule (16) or rule (15). In the case of rule (15) the premise that $\forall (s, t_2) \in \Sigma_1$ is trivially true.

1.2° Let a be an action of T_1 that is not shared with T_2, or similarly $a \in \mathit{act}_{S_1} \setminus \mathit{act}_{S_2}$. We want to apply rule (13) and want to show that the premises are fulfilled. The first two premises are fulfilled by the case that we are looking at. The third premise is fulfilled by the following argument. Because t'_1 is making a step we have that $(t_1, t_2) \xrightarrow{a}{}^{T_1 \cdot T_2}_{\square} (t'_1, t_2)$. By the definition of states_{S_1} and R_1 we have that $(s, (t_1, t_2)) \in R$ for every pair $(s, t_1) \in \Sigma_1$. Because R is a modal refinement of T by S we have that $s \xrightarrow{a}{}^{S}_{\square} s'$ and $(s', (t'_1, t_2)) \in R$ for every pair $(s, t_1) \in \Sigma_1$. The third premise will trivially hold and we can even conclude that Σ'_1 will be nonempty. Now we can apply rule (13) and we can conclude that indeed $(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\diamond} (((\mathit{act}_{T_2} \setminus \mathit{act}_{T_1})\text{-}\mathit{closure}(\Sigma'_1), t'_1)$. From this we can conclude that a similar must transition exists because the two transition relations are identical. Finally we can conclude that $(((act_{T_2}\backslash act_{T_1})\text{–}closure(\mathit{\Sigma}'_1), t'_1) \in R_1$ because the generated transitions stay within $states_{S_1}$ and $t'_1 \in states_{T_1}$.

1.3° Let a be an action of T_1 that is shared with T_2, or similarly $a \in act_{S_1} \cap act_{S_2}$. We want to apply rule (14) and (15), in two different sub cases, and want to show that the premises are fulfilled. The first two premises of both rules are fulfilled by the case that we are looking at. The third premise of rule (14) and (15) are each others opposites, such that the one is true when the other is false and vise versa. Looking at the case where $\exists (s, t_2) \in \mathit{\Sigma}_1 . s \xrightarrow{a}{}^{S}_{\Box}$, which is exactly the third premise of rule (14), then we can conclude that the last premise for rule (14) is true by the following argument. Because S is consistent we know that there is a transition $s \xrightarrow{a}{}^{S}_{\Diamond}$. Because R is a modal refinement of T by S and we can conclude that the only way that this transition can exist is if a similar transition $t_2 \xrightarrow{a}{}^{T_2}_{\Diamond} t'_2$ exists such that $(t_1, t_2) \xrightarrow{a}{}^{T_1 \cdot T_2}_{\Diamond}$. The fourth premise of rule (14) is trivially true, but we can now conclude that $\mathit{\Sigma}'_1$ is nonempty. Now we can apply rule (14) and we can conclude that indeed $(\mathit{\Sigma}_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (((act_{T_2} \setminus act_{T_1})\text{–}closure(\mathit{\Sigma}'_1), t'_1)$. From this we can conclude that a similar must transition exists because the two transition relations are identical. Finally we can conclude that $(((act_{T_2} \setminus act_{T_1})\text{–}closure(\mathit{\Sigma}'_1), t'_1) \in R_1$ because the generated transitions stay within $states_{S_1}$ and $t'_1 \in states_{T_1}$.

Now turning to the other sub case where $\forall (s, t_2) \in \mathit{\Sigma}_1 \; s \not\xrightarrow{a}{}^{S}_{\Box}$. In this case there are no must transitions in S requiring the behavior but S_1 will have the behavior because T_1 requires it. From this we can conclude that $(\emptyset, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\emptyset, t'_1)$ and that a similar must transition exists because the two transition relations are identical. Finally we can conclude that $(((act_{T_2} \backslash act_{T_1})\text{–}closure(\emptyset), t'_1) \in R_1$ because the generated transitions stay within $states_{S_1}$ and $t'_1 \in states_{T_1}$.

This finishes one direction of the proof. Lets now consider a may transition $(\mathit{\Sigma}_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\mathit{\Sigma}'_1, t'_1)$. We need to show that a transition $t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t'_1$ exists such that $((\mathit{\Sigma}'_1, t'_1) \in R_1)$

1.4° This transition could have been generated by one of the four rules (13)-(16). In two of the cases we can directly conclude that a transition $t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t'_1$ exists. In the other two cases we can conclude that this transition exists because the rules require a similar must transition and T_1 is syntacticly consistent. Now it follows directly from Lemma 25 that $(\mathit{\Sigma}'_1, t'_1) \in R_1$

2° The proof that $S_2 \leq_{\mathrm{m}} T_2$ is entirely symmetric to the proof that $S_1 \leq_{\mathrm{m}} T_1$.

3° Show that $S_1 \otimes S_2 \leq_{\mathrm{m}} S$. We do that by arguing that

$$\begin{aligned} R_2 = \{(((\varSigma_1, t_1), (\varSigma_2, t_2)), s) \mid \\ ((act_{T_1} \setminus act_{T_2})\text{-}closure(s, t_1) \subseteq \varSigma_2 \text{ and} \\ ((act_{T_2} \setminus act_{T_1})\text{-}closure(s, t_2) \subseteq \varSigma_1 \text{ and} \\ \varPi_1(\varSigma_1) \text{ is observationally consistent wrt } act_{T_2} \setminus act_{T_1} \text{ and} \\ \varPi_1(\varSigma_2) \text{ is observationally consistent wrt } act_{T_1} \setminus act_{T_2}\} \end{aligned} \tag{18}$$

is a modal refinement of S by $S_1 \otimes S_2$. First we should argue that

$$((start_{S_1}, start_{S_2}), start_S) \in R_2 \ \ . \tag{19}$$

Obviously

$$(act_{T_2} \setminus act_{T_1})\text{-}closure(start_S, start_{T_2}) \subseteq \varSigma_1^0 \text{ and} \tag{20}$$

$$(act_{T_1} \setminus act_{T_2})\text{-}closure(start_S, start_{T_1}) \subseteq \varSigma_2^0 \tag{21}$$

(actually equalities hold). Observational consistency of projections of $\varSigma_1^0$ and $\varSigma_2^0$ follows from consistency of S, T_1, T_2 and Lemma 24.

We shall discuss that the may transition relation preserves the refinement. Take any $(((\varSigma_1, t_1), (\varSigma_2, t_2)), s) \in R_2$ and a transition step

$$((\varSigma_1, t_1), (\varSigma_2, t_2)) \xrightarrow{a}{}^{S_1 \otimes S_2}_{\Diamond} ((\varSigma_1', t_1'), (\varSigma_2', t_2')) \tag{22}$$

We want to find a state s' such that $s \xrightarrow{a}{}^{S}_{\Diamond} s'$ and $(((\varSigma_1', t_1'), (\varSigma_2', t_2')), s') \in R_2$. Note that due to the way R_2 is constructed we know that neither $\varSigma_1$ nor $\varSigma_2$ are empty. The transition step of the composition must then be created by both components taking a shared action (and both following rule (14)) or by one component taking a non-shared action, by rule (13), and the other not changing state.

Observe that rule (16), can never give rise to such a transition as it would require $\varSigma_1$ or $\varSigma_2$ to be empty, which we have just ruled out.

3.1° Let $a \in act_{S_1} \cap act_{S_2}$. We want to first argue that both components take steps generated by rule (14) and not rule (15). The latter would require that either t_1 or t_2 enjoys a must transition $t_i \xrightarrow{a}{}^{T_i}_{\Box} t_i'$. If both transitions existed, they would imply that also $s \xrightarrow{a}{}^{S}_{\Box} s'$ (since $(s, (t_1, t_2)) \in R$, S is deterministic), which would contradict the joint premises of the rules. So only one of the two must transitions can exist. But then the other component is taking a transition generated by rule (14) implying that $s \xrightarrow{a}{}^{S}_{\Box} s'$, contradicting premises of rule (15) (for both components). In other words rule (15) could not have been used, so for some sets $\varSigma_1''$, $\varSigma_2''$:

$$(\varSigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} ((act_{T_2} \setminus act_{T_1})\text{-}closure(\varSigma_1''), t_1') \tag{23}$$

$$(\varSigma_2, t_2) \xrightarrow{a}{}^{S_2}_{\Diamond} ((act_{T_1} \setminus act_{T_2})\text{-}closure(\varSigma_2''), t_2') \tag{24}$$

From that we derive that rule (14) must have been used to create both of these transitions, which implies that there exists $(s_1, p_2) \in \Sigma_1$ such that $s_1 \xrightarrow{a}{}^{S}_{\Box} s'_1$ for some state s'_1. Since $\Pi_1(\Sigma_1)$ is an observationally consistent set with respect to $act_{T_2} \setminus act_{T_1}$ then there exists a state s' such that $s \xrightarrow{a}{}^{S}_{\Diamond} s'$ and (s'_1, s') is an observationally consistent pair of states. Since S is deterministic the same argument can be used for all elements in $\Pi_1(\Sigma''_1)^5$, which with help of Lemmas 23 and 24 leads us to a conclusion that the first component of $(act_{T_2} \setminus act_{T_1})$–$closure(\Sigma''_1)$ is observationally consistent wrt $(act_{T_2} \setminus act_{T_1})$.

Since rule (14), or more precisely its counterpart for S_2, must have been used to construct transition (24) we can also conclude that $t_2 \xrightarrow{a}{}^{T_2}_{\Diamond} t'_2$. So by premises of rule (14) instantiated for transition (23) we conclude that $(s', t'_2) \in \Sigma''_1$ and hence is in the closure.

Symmetric arguments can be used to argue that the first component of the closure of Σ''_2 is observationally consistent wrt $act_{T_1} \setminus act_{T_2}$, and that $(s', t'_1) \in \Sigma''_2$ and hence also in its closure, which finishes the proof of this case.

3.2° Let $a \in act_{S_1} \setminus act_{S_2}$. Then we know that:

$$(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\Sigma'_1, t'_1) \text{ and } \Sigma'_2 = \Sigma_2 \text{ and } t'_2 = t_2 \ . \tag{25}$$

It easy to conclude that the step of T_1 has been generated by rule (13) and not rule (16) (we have already argued against this case above: $\Sigma_1 \neq \emptyset$).

The fact that (Σ_1, t_1) is able to make an a step by rule (13) implies that some state of s paired with some state of T_2 in Σ_1 requires such a step. By observational consistency of $\Pi_1(\Sigma_1)$ we have that necessarily $s \xrightarrow{a}{}^{S}_{\Diamond} s'$ for some s'. Moreover $(s', t_2) \in \Sigma'_1$ (by rule (13)) and $(s', t'_1) \in \Sigma_2$ since $(s', t'_1) \in (act_{T_1} \setminus act_{T_2})$–$closure(s, t_1) = \Sigma_2$. Since Σ_2 does not change, there is no need to argue for its consistency. Consistency of $\Pi_1(\Sigma'_1)$ follows from the fact that a transition is taken, which cannot move outside the consistent set (a hidden must transition).

3.3° The case when s takes a transition over a non-shared action of S_2 is entirely symmetric.

Observe that implicitly (by analyzing all interaction possibilities) we have ruled out a possibility of a deadlock between S_1 and S_2.

Let us now turn towards the must transition relations. Assume that for some action a and state s' we have that $s \xrightarrow{a}{}^{S}_{\Box} s'$.

3.4° Let $a \in act_{T_1} \cap act_{T_2}$. Since $(s, (t_1, t_2)) \in R$ and S is syntactically consistent, we get that $(t_1, t_2) \xrightarrow{a}{}^{T_1 \cdot T_2}_{\Diamond} (t'_1, t'_2)$ for some t'_1, t'_2 and further that $t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t'_1$

[5] In the nondeterministic case we would probably have to extend the definition of observational consistency with a universal quantifier, instead of the existential, which it is using now.

and $t_2 \xrightarrow{a}{}^{T_2}_{\Diamond} t'_2$. But these imply by rule (14) that $(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\Sigma'_1, t'_1)$, where $(act_{T_2} \backslash act_{T_1})$–*closure*$(s', t'_2) \subseteq \Sigma'_1$ and similarly $(\Sigma_2, t_2) \xrightarrow{a}{}^{S_2}_{\Diamond} (\Sigma'_2, t'_2)$, where $(act_{T_1} \backslash act_{T_2})$–*closure*$(s', t'_1) \subseteq \Sigma'_2$.

We have chosen that the must transition relations of both S_1 and S_2 are identical with their respective may transition relations, so we can conclude that $((\Sigma_1, t_1), (\Sigma_2, t_2)) \xrightarrow{a}{}^{S_1 \otimes S_2}_{\Box} ((\Sigma'_1, t'_1), (\Sigma'_2, t'_2))$.

Observational consistency of the first components of Σ'_1 and Σ'_2 can be argued as in earlier cases (existence of a single must transition of s guarantees that none of s transitions labeled in a and sourced in states of Σ_i can leave outside the set of consistent states).

3.5° Let $a \in act_{T_1} \backslash act_{T_2}$. Since $(s, (t_1, t_2)) \in R$ and S is syntactically consistent, we get that $(t_1, t_2) \xrightarrow{a}{}^{T_1 \cdot T_2}_{\Diamond} (t'_1, t_2)$ and further that $t_1 \xrightarrow{a}{}^{T_1}_{\Diamond} t'_1$. But this implies by rule (13) that $(\Sigma_1, t_1) \xrightarrow{a}{}^{S_1}_{\Diamond} (\Sigma'_1, t'_1)$, where $(act_{T_2} \backslash act_{T_1})$–*closure*$(s', t_2) \subseteq \Sigma'_1$. Also $(act_{T_1} \backslash act_{T_2})$–*closure*$(s', t'_1) \subseteq \Sigma_2$ since the transition performed by this pair is within the original closure $(act_{T_1} \backslash act_{T_2})$–*closure*$(s, t_1)$, which was a subset of Σ_2.

As we have chosen that must transition relation of S_1 is identical with its may transition relation, we can conclude that:

$$((\Sigma_1, t_1), (\Sigma_2, t_2)) \xrightarrow{a}{}^{S_1 \otimes S_2}_{\Box} ((\Sigma'_1, t'_1), (\Sigma_2, t_2)) \ . \tag{26}$$

Finally Σ'_1 is observationally consistent as s only takes a hidden transition here (with respect to the set of ignored actions), which finishes the proof for this case.

3.6° The case where S_2 takes an independent step is symmetric. □

Observe that the above theorem can be used to generate decompositions of simulations and bisumulations (which are special cases of modal refinement).

Proof (Thm. 15). Show that

$$R_3 = \{((i_1, i_2), (t_1, t_2)) \in states_{I_1 \cdot I_2} \times states_{T_1 \cdot T_2} \mid i_1 \leq_{\mathrm{m}} t_1 \wedge i_2 \leq_{\mathrm{m}} t_2\} \tag{27}$$

is a modal refinement of $T_1 \cdot T_2$ by $I_1 \cdot I_2$.

1° Consider $(i_1, i_2) \xrightarrow{a}_{\Diamond} (i'_1, i'_2)$. We have to consider four cases: 1.1° $a \in ext_{I_1 \cdot I_2}$, $i_1 \xrightarrow{a}_{\Diamond} i'_1$ and $i_2 = i'_2$. As $i_1 \leq_{\mathrm{m}} t_1$ there must exist a t'_1 such that $t_1 \xrightarrow{a}_{\Diamond} t'_1$ and $i'_1 \leq_{\mathrm{m}} t'_1$, so $((i'_1, i_2), (t'_1, t_2)) \in R_3$. By definition of the composition operator $(\cdot)$ we get that $(t_1, t_2) \xrightarrow{a}_{\Diamond} (t'_1, t_2)$: the only possibility for it could not hold is when (t'_1, t_2) has been pruned in $T_1 \cdot T_2$, so there exists a sequence of internally controllable must transitions leading from (t'_1, t_2) to an error state (t''_1, t''_2) where $t''_k \xrightarrow{a!}_{\Diamond} t'''_k$ and $t''_{3-k} \not\xrightarrow{a?}_{\Diamond}$, where $k \in 1, 2$. But then a corresponding sequence would exist in $I_1 \cdot I_2$, meaning that $(i_1, i_2) \xrightarrow{a}_{\Diamond} (i'_1, i_2)$ was not possible to begin with (also pruned). Finally it is easy to see $((s'_1, s'_2), (t'_1, t'_2)) \in R_3$.

1.2° $a \in ext_{I_1 \cdot I_2}$, $i_2 \xrightarrow{a}_{\Diamond} i'_2$ and $i_1 = i'_1$ is symmetric.

1.3° $a \in int_{I_1 \cdot I_2}$, $i_1 \xrightarrow{a!}_{\Diamond} i'_1$ and $i_2 \xrightarrow{a?}_{\Diamond} i'_2$. Then by $i_1 \leq_{\mathrm{m}} t_1$ and $i_2 \leq_{\mathrm{m}} t_2$ we conclude that there exists t'_1, t'_2 such that $t_1 \xrightarrow{a!}_{\Diamond} t'_1$ and $t_2 \xrightarrow{a?}_{\Diamond} t'_2$ and $i'_1 \leq_{\mathrm{m}} t'_1$ and $i'_2 \leq_{\mathrm{m}} t'_2$. By definition of the composition operator $(\cdot)$ we get that $(t_1, t_2) \xrightarrow{a}_{\Diamond} (t'_1, t'_2)$: the only possibility for it could not hold is when (t'_1, t'_2) has been pruned in $T_1 \cdot T_2$, so there exists a sequence of internally controllable must transitions leading from (t'_1, t'_2) to an error state (t''_1, t''_2) where $t''_k \xrightarrow{a!}_{\Diamond} t'''_k$ and $t''_{3-k} \not\xrightarrow{a?}_{\Diamond}$, where $k \in 1, 2$. But then a corresponding sequence would exist in $I_1 \cdot I_2$, meaning that $(i_1, i_2) \xrightarrow{a}_{\Diamond} (i'_1, i'_2)$ was not possible to begin with. Finally it is easy to see $((s'_1, s'_2), (t'_1, t'_2)) \in R_3$.

1.4° $a \in int_{I_1 \cdot I_2}$, $i_2 \xrightarrow{a!}_{\Diamond} i'_2$ and $i_1 \xrightarrow{a?}_{\Diamond} i'_1$. The argument follows as in 1.3°.

2° Consider $(t_1, t_2) \xrightarrow{a}_{\Box} (t'_1, t'_2)$. We have four subcases again out of which 2 are interesting.

2.1° $a \in ext_{T_1 \cdot T_2}$ and $t_1 \xrightarrow{a}_{\Box} t'_1$ and $t_2 = t'_2$. Then by $i_1 \leq_{\mathrm{m}} t_1$ there exist i'_1 such that $i_1 \xrightarrow{a}_{\Box} i'_1$ and $i'_1 \leq_{\mathrm{m}} t'_1$. By similar argument as above $(i_1, i_2) \xrightarrow{a}_{\Box} (i'_1, i_2)$ (because if (i'_1, i_2) was pruned then so was (i_1, i_2), for which we assumed that it was not) and $(i'_1, i'_2), (t'_1, t'_2) \in R_3$.

2.2° $a \in ext_{T_1 \cdot T_2}$ and $t_2 \xrightarrow{a}_{\Box} t'_2$ and $t_1 = t'_1$. Argument as above.

2.3° $a \in int_{T_1 \cdot T_2}$ and $t_1 \xrightarrow{a!}_{\Box} t'_1$ and $t_2 \xrightarrow{a?}_{\Box} t'_2$. Then by $i_1 \leq_{\mathrm{m}} t_1$ and $i_2 \leq_{\mathrm{m}} t_2$ there exist i'_1 and i'_2 such that $i_1 \xrightarrow{a}_{\Box} i'_1$ and $i_2 \xrightarrow{a}_{\Box} i'_2$ and $i'_1 \leq_{\mathrm{m}} t'_1$ and $i'_2 \leq_{\mathrm{m}} t'_2$. By a similar argument involving the definition of $(\cdot)$ as above we get $(i_1, i_2) \xrightarrow{a}_{\Box} (i'_1, i'_2)$ (as if (i'_1, i'_2) then so would (i_1, i_2) which was assumed not to be pruned). So $((i'_1, i'_2), (t'_1, t'_2)) \in R_3$, which finishes the proof. □