Universität Augsburg

Interface Automata with Error States

Ferenc Bujtor and Walter Vogler

Report 2012-09 July 2012

Institut für Informatik
D-86135 Augsburg

Interface-Automata with Error States

Ferenc Bujtor and Walter Vogler

Institut für Informatik, Universität Augsburg, D–86135 Augsburg, Germany
bujtor@informatik.uni-augsburg.de vogler@informatik.uni-augsburg.de

Abstract. De Alfaro and Henzinger advocated interface automata to model and study behavioural types, which describe communication patterns of systems while abstracting e.g. from data. They come with a specific parallel composition: if, in some state, one component tries to make an output, which the other one cannot receive, the state is regarded as an error. Error states are removed along with some states leading to them. As refinement relation an alternating simulation is introduced.
In this report, we study to what degree this refinement relation is justified by the desires to avoid error states and to support modular refinement. For this, we leave the error states in place and mark them as such instead of removing them in the composition. Our Error-I-O-Transition systems are slightly more general than Interface automata, which are restricted to input determinism.
Our basic requirement is: an implementation must be error free, if the specification is. For two different notions of error freeness, we determine the coarsest precongruences contained in the respective basic refinement relations. We characterize these best refinement relations meeting our desirables with trace sets. Thus our precongruences are less discriminating than simulation-based ones. Along the way we point out an error in an early paper by de Alfaro and Henzinger.

1 Introduction

Interface automata as advocated by de Alfaro and Henzinger e.g. in [4] give an abstract description of the communication behaviour of a system specification or process in terms of input and output actions. Based on this behavioural type, one can study whether two systems are compatible if put in parallel, and one can define a refinement for specifications. Essential for such a setting is that the refinement relation is a precongruence for parallel composition; in particular, if we refine two compatible specifications, it must be guaranteed that the refined specifications are compatible again.

Two processes composed in parallel synchronize on their common actions. Since interaction is supposed to be binary, such a common action has to be an output of one process and an input of the other; after synchronization, the action is internalized, i.e. hidden from the environment. Outputs are under the control of the respective process, so the process will not wait for the other one when performing an output. Now the basic idea for compatibility is the following: if, in a state of a parallel composition, one of the two processes tries to synchronize by performing an output that the other process should but cannot receive, this state is regarded as catastrophic; since the second process is not ready to receive the output, it might malfunction – such an error state has to be avoided. Observe that interface automata are not input enabled as required for the I,O-automata of [8]. Instead, a missing input in a state corresponds to the requirement that a prospective environment must not send this input to this state.

There are two essential design decisions in the approach of [4] that we will scrutinize in this paper. First, this approach follows an optimistic view: an error state in a parallel composition is no problem, if it cannot be reached in a helpful environment. This is reflected in the precise definition of parallel composition: first a more standard composition is determined; then all states are removed that can reach an error state just by locally controlled, i.e. output and internal actions (so-called output pruning). This way, also the last input before reaching an error state is forbidden since its target state is removed. Although this definition has some intuitive justification, its details appear somewhat out of the blue; e.g. the authors of [1] prefer a pessimistic view where every reachable error state is a problem. The second decision to take some alternating simulation as refinement relation also seems somewhat arbitrary. Actually, the same authors used a slightly different relation for a slightly larger class of automata in the earlier [3]; no real argument is given for the change.

In this paper, we will work out to what degree these design decisions can be justified from some more basic and, hopefully, more agreeable ideas. We consider processes modelled by labelled transition systems (LTSs) with disjoint input and output actions and an internal action τ, more or less like the interface automata of [4]. Since we try not to exclude any possibilities prematurely, our LTS have explicit error states; we call them Error-IO Transition Systems or EIO for short. We consider a standard definition of parallel composition where additionally error states occur as described above; a composed system also reaches an error state if one of the components reaches one.

An undisputable requirement for a refinement relation is that an error-free specification should only be refined by an error-free system. This can be understood as a basic refinement relation, which is parametric in the exact meaning of error-free: in the optimistic view, error-free means that no error state can be reached by locally controlled actions only; in the pessimistic view, a system would be error-free only if no error state is reachable at all. In this paper we study the optimistic and a hyper-optimistic meaning of error freeness, while considerations of the pessimistic variant have not been finished yet.

For modular reasoning, which is at the heart of the approach under study, the refinement relation must be a precongruence: if a component of a parallel composition is replaced by a refinement, the composition itself gets refined. Our basic relations fail to be precongruences. Therefore, in each case, we will characterize the precongruence that is fully abstract w.r.t. the respective basic relation and parallel composition, i.e. we will determine the coarsest precongruence for parallel composition that is contained in the basic relation. These precongruences are thus justified by very basic requirements.

It turns out that, in the optimistic case, the precongruence can be characterized as (componentwise) inclusion for a pair of trace sets. Such a pair of sets can best be obtained from a given EIO by performing output pruning on it. Phrased another way, with this precongruence each EIO is equivalent to one without error states – provided that the initial state is not pruned. Essentially, we can work with EIO without error states, i.e. with interface automata and with the parallel composition of [4]. Whenever output pruning removes the initial state of a composition, the components are called incompatible. Thus, as in [4], only compatible systems should be composed, and refining compatible specifications leads indeed to compatible systems.

While this justifies the first design decision in [4], our precongruence shows that alternating simulation is unnecessarily strict. In fact, our precongruence is not really new. A setting with input and outputs where unexpected inputs lead to errors has been studied long before [4] for speed-independent (thus asynchronous) circuits. In this context, essentially the same pair of trace sets has been suggested by Dill in [5]. The difference is that Dill does not start from an operational model as we do, but on a semantic level with pairs of trace sets; he requires these pairs to be input enabled. On this semantic level, he also uses output pruning; a normalized form of his pairs coincides with our pairs (which are in a sense input enabled, although missing inputs in the EIO are so essential). Dill uses transition systems for graphical representation, but (if we are not mistaken) with one exception all of these are deterministic; parallel composition is never performed on transition systems but on pairs of trace sets only.

There is a subtle point about output pruning. Interface automata in [4] are deterministic w.r.t. input actions. Since we do not require this here, our output pruning is a bit different from the one in [4]. In fact, the interface automata in [3] are not input deterministic, but output pruning used there is the same as the one in [4]. As a consequence, Theorem 1 of [3] claiming associativity for parallel composition is wrong. In the settings of [3, 4], associativity is tricky; in our setting with error states, it is dead easy.

It might seem that we have actually prescribed output pruning in our optimistic approach: we consider only locally reachable errors as relevant and output pruning removes exactly those states that can reach an error locally. To consider an alternative, we turn to a 'hyper-optimistic' approach next, where only internally reachable errors are relevant. With this more generous notion of error freeness we obtain a slightly stronger precongruence; but it is still based on output pruning, with the difference that some information about the removed outputs must be retained. This is also a feasible precongruence but, compared to our first one, it looks unnecessarily involved technically.

Currently we are working on the pessimistic approach.

2 Definitions and Notation

First we define our scenario. We use labelled transition systems (LTS) with disjoint input and output actions. The systems can also perform internal, unobservable actions, denoted by τ; in interface automata, internal actions have different names, but semantically these never play a rôle. Additionally, the LTS has a separate set of error states; such states can be created in a parallel composition.

Definition 1 (Error-IO-Transition-System). An *Error-IO-Transition-System (EIO)* is defined as a tuple $S = (Q, I, O, \delta, q_0, E)$, where

- Q – a set of states
- I, O – disjoint sets of input and output actions
- $\delta \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$ – a transition relation
- $q_0 \in Q$ – an initial state
- $E \subseteq Q$ – a set of *error* states

We define the *actions* of S by $\Sigma := I \cup O$, and its signature by $Sig(S) = (I, O)$.

As a shorthand we write Q_1, I_1, δ_1 etc. for components of the EIOs S_1 and Q_2, I_2, δ_2 etc. for S_2 and so on. We also use this notation for semantics like ET_1 for $ET(S_1)$ as defined later on. We also write $q \xrightarrow{a} p$ for $(q, a, p) \in \delta$ and $q \xrightarrow{a}$ for $\exists p : (q, a, p) \in \delta$. We extend this to sequences $w \in (\Sigma \cup \{\tau\})^*$, writing $q \xrightarrow{w} p$, $(q \xrightarrow{w}\ )$ whenever $q \xrightarrow{a_1}\xrightarrow{a_2} \cdots \xrightarrow{a_n} p$, $(q \xrightarrow{a_1}\xrightarrow{a_2} \cdots \xrightarrow{a_n})$ with $w = a_1 \cdots a_n$. Furthermore we define $w|_B$ as the projection of an action sequence to a set of actions B. We define $q \overset{w}{\Rightarrow} p$ for $w \in \Sigma^*$ by $q \overset{w}{\Rightarrow} p$ if $\exists w' : w'|_\Sigma = w \wedge q \xrightarrow{w'} p$. A sequence $q_0 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots q_n$ is a *run underlying* $a_1 \dots a_{n-1}|_\Sigma$.

In a parallel composition, all common actions are first synchronized without hiding, and then they are hidden. Two EIOs can only be composed, if their input and output actions fit together, i.e. the EIOs have neither common inputs nor common outputs.

Definition 2 (Parallel Composition). Two EIOs S_1, S_2 are *composable* if $I_1 \cap I_2 = \emptyset = O_1 \cap O_2$. The *parallel composition without hiding* is defined for two composable EIOs as $S_1 \parallel S_2 = (Q, I, O, \delta, q_0, E)$, where

- $Q = (Q_1 \times Q_2)$
- $I = \big((I_1 \backslash O_2) \cup (I_2 \backslash O_1)\big)$
- $O = \big(O_1 \cup O_2\big)$
- $q_0 = (q_{01}, q_{02})$

Furthermore, with $Synch(S_1, S_2) = (I_1 \cap O_2) \cup (I_2 \cap O_1)$ being the set of synchronized actions, we define

- $$\begin{aligned}\delta =&\{\big((q_1, q_2), \alpha, (p_1, q_2)\big) \mid (q_1, \alpha, p_1) \in \delta_1, \alpha \in (\Sigma_1 \cup \{\tau\}) \backslash Synch(S_1, S_2)\} \cup \\ &\{\big((q_1, q_2), \alpha, (q_1, p_2)\big) \mid (q_2, \alpha, p_2) \in \delta_2, \alpha \in (\Sigma_2 \cup \{\tau\}) \backslash Synch(S_1, S_2)\} \cup \\ &\{\big((q_1, q_2), \alpha, (p_1, p_2)\big) \mid (q_1, \alpha, p_1) \in \delta_1, (q_2, \alpha, p_2) \in \delta_2, \alpha \in Synch(S_1, S_2)\}\end{aligned}$$

- $$\begin{aligned}E =&(Q_1 \times E_2) \cup (E_1 \times Q_2) && \text{‘inherited errors’} \\ &\cup \{(q_1, q_2) \mid \exists a \in O_1 \cap I_2 : q_1 \xrightarrow{a} \wedge q_2 \stackrel{a}{\not\rightarrow}\} \\ &\cup \{(q_1, q_2) \mid \exists a \in I_1 \cap O_2 : q_1 \stackrel{a}{\not\rightarrow} \wedge q_2 \xrightarrow{a}\} && \text{‘new errors’}\end{aligned}$$

The *parallel composition (with hiding)* $S_1 \mid S_2$ differs only in the definition of its outputs and its transition function.

- $O = \big((O_1 \backslash I_2) \cup (O_2 \backslash I_1)\big)$

$$\begin{aligned}\delta =&\{\big((q_1,q_2),\alpha,(p_1,q_2)\big) \mid (q_1,\alpha,p_1) \in \delta_1, \alpha \in (\Sigma_1 \cup \{\tau\}) \backslash Synch(S_1,S_2)\} \cup \\ &\{\big((q_1,q_2),\alpha,(q_1,p_2)\big) \mid (q_2,\alpha,p_2) \in \delta_2, \alpha \in (\Sigma_2 \cup \{\tau\}) \backslash Synch(S_1,S_2)\} \cup \\ &\{\big((q_1,q_2),\tau,(p_1,p_2)\big) \mid (q_1,\alpha,p_1) \in \delta_1, (q_2,\alpha,p_2) \in \delta_2, \alpha \in Synch(S_1,S_2)\}\end{aligned}$$

Again we introduce a shorthand S_{12} for $S_1 \mid S_2$ and use it accordingly for its components and semantics.

For our results and proofs, we also define $\mid$ and $\|$ as parallel composition on traces with and without hiding respectively.

Definition 3 (Parallel Composition on Traces). Given two composable EIOs S_1, S_2, $w_1 \in \Sigma_1^*, w_2 \in \Sigma_2^*, W_1 \subseteq \Sigma_1$ and $W_2 \subseteq \Sigma_2$, we define

- $w_1 \| w_2 = \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid w|_{\Sigma_1} = w_1 \wedge w|_{\Sigma_2} = w_2\}$
- $w_1 \mid w_2 = \{w|_{\Sigma_{12}} \mid w \in w_1 \| w_2\}$
- $W_1 \| W_2 = \bigcup\{w_1 \| w_2 \quad \mid \quad w_1 \in W_1 \wedge w_2 \in W_2\}$
- $W_1 \mid W_2 = \bigcup\{w_1 \mid w_2 \quad \mid \quad w_1 \in W_1 \wedge w_2 \in W_2\}$

We will base our semantics on traces that can lead to error states. In this context, we will use a pruning function, which removes all output actions from the end of a trace. We also define a function for arbitrary continuation of traces; for trace sets, this generalizes to describing the continuation or suffix closure.

Definition 4 (Pruning and Continuation Functions). Given an EIOS, we define

- $prune: \ \Sigma^* \to \Sigma^*$, $w \mapsto u$, where $w = uv$, $u = \varepsilon \vee u \in \Sigma^* \cdot I$ and $v \in O^*$
- $cont: \ \Sigma^* \to \mathfrak{P}(\Sigma^*)$, $w \mapsto \{wu \mid u \in \Sigma^*\}$
- $cont: \ \mathfrak{P}(\Sigma^*) \to \mathfrak{P}(\Sigma^*)$, $L \mapsto \{cont(w) \mid w \in L\}$

For composable EIOs S_1 and S_2, consider a run of their parallel composition $S_1 \| S_2$ that justifies statement $(q_1,q_2) \overset{w}{\Rightarrow} (p_1,p_2)$ for $w \in \Sigma^*$. It is well known and not difficult to see that such a run can be projected to runs of S_1 and S_2, passing through all the first, second resp., components of the states of the composed run. These *projected runs* justify $q_i \overset{w_i}{\Rightarrow} p_i$ with $w|_{\Sigma_i} = w_i, \ i = 1,2$. Vice versa, any two runs of S_1 and S_2 justifying $q_i \overset{w_i}{\Rightarrow} p_i$ with $w|_{\Sigma_i} = w_i, \ i = 1,2$, are projections of a unique run of $S_1 \| S_2$ justifying $(q_1,q_2) \overset{w}{\Rightarrow} (p_1,p_2)$. From this, the first claim of the next Lemma follows.

Each run of $S_1 \| S_2$ corresponds to one of $S_1 \mid S_2$ – simply replace some actions by τ. We also call the projected runs of the former the *projections* of the latter.[1] In such a case, we also say that the $q_i \overset{w_i}{\Rightarrow} p_i$, $i = 1,2$, are the *projections* of $(q_1,q_2) \overset{w'}{\Rightarrow} (p_1,p_2)$ in $S_1 \mid S_2$, where $w' \in w_1 \mid w_2$. These considerations justify the second claim of the next Lemma.

[1] This is a slight abuse of language, since these projections have additional actions and are not really unique; the possible differences do not matter.

Lemma 5 (Basic Language of Composition) *For two composable EIOs S_1 and S_2, we have*

1. $L(S_1 \parallel S_2) = L(S_1) \parallel L(S_2)$
2. $L(S_1 \mid S_2) = L(S_1) \mid L(S_2)$.

3 Local Errors

We are now ready to consider some basic relations for refinement of a specification. We will use variations of the notation '$Impl \sqsubseteq^B Spec$' to denote that $Impl$ in some basic sense is an implementation of, i.e. refines, the specification $Spec$. Throughout, we require $Impl$ and $Spec$ to have the *same signature*.

3.1 Precongruence for Local Errors

In this section, we will start with a variant based on 'local actions', which are all internal and output actions. We consider the following requirement: An implementation can only have an error state reachable by local actions if the specification does so as well. This is an optimistic view: It only considers processes to be dangerous, if they can run into an error on their own, i.e. using only local actions. Formally:

Definition 6 (Local Basic Relation). An *error is locally reachable* in an EIOS, if $\exists w \in O^* : w \in StT(S)$. We have $Impl \sqsubseteq^B_{loc} Spec$, whenever an error is locally reachable in $Impl$ only if an error is locally reachable in $Spec$.

We denote the fully abstract precongruence with respect to $\sqsubseteq^B_{loc}$ and $\mid$ by $\sqsubseteq^c_{loc}$.

In order to find the coarsest precongruence, we will need several trace sets for characterizing the new precongruences. An EIO can reach an error state with a strict error trace w; if in a parallel composition the environment provides the inputs and accepts the outputs in w, the composition will reach an error state as well. If an EIO can perform a trace w such that input a is not possible in the state reached, then wa is a missing input trace; again, if an environment provides the inputs and accepts the outputs in wa, the composition will reach an error state when the environment produces a.

Definition 7 (Error Traces). We define the following trace sets for an EIO S:

- *strict error traces*: $StT(S) = \{w \in \Sigma^* \mid q_0 \stackrel{w}{\Rightarrow} q \in E\}$
- *pruned error traces*: $PrT(S) = \{prune(w) \mid w \in StT(S)\}$
- *missing input traces*: $MIT(S) = \{wa \in \Sigma^* \mid q_0 \stackrel{w}{\Rightarrow} q \wedge a \in I \wedge q \stackrel{a}{\not\rightarrow}\}$
- *basic language*: $L(S) = \{w \in \Sigma^* \mid q_0 \stackrel{w}{\Rightarrow}\}$

Now the coarsest precongruence can be characterized with the following local error semantics; the intuitions are as follows. Errors arise in a composition because a component performs a strict error trace or because it cannot accept some input after a trace; in the first case, the error is already unavoidable if the error state can be reached by local actions only. These ideas justify our interest in traces that lead to errors and the use of PrT and MIT in the definition of ET below. But as already explained above, the environment must take part in such problematic behaviour, hence we are also interested in the basic language of a system.

If *along* a trace an error can occur, it does not matter anymore whether the trace itself leads to an error state or whether it can be performed at all. Thus, we want to obliterate this information about the trace itself; for this purpose, we close the set of problematic traces under continuation, and we include this extended set in the language; this technique of flooding is well known e.g. in the context of failures semantics [2].

It will turn out that we can characterize $\sqsubseteq^c_{loc}$ as componentwise set inclusion for pairs $(ET(S), EL(S))$; we introduce a sign for this relation.

Definition 8 (Local Error Semantics). Let S be an EIO.

- The set of *error traces* of S is $ET(S) = cont(PrT(S)) \cup cont(MIT(S))$;
- the *flooded language* of S is $EL(S) = L(S) \cup ET(S)$.

For two EIOs $Impl$ and $Spec$ with the same signature, we write

$$Impl \sqsubseteq_{loc} Spec \text{ if } ET(Impl) \subseteq ET(Spec) \text{ and } EL(Impl) \subseteq EL(Spec)$$

and we call $Impl$ and $Spec$ *local-error equivalent* if $Impl \sqsubseteq_{loc} Spec$ and $Spec \sqsubseteq_{loc} Impl$.

For the characterization result, it is crucial that the local error semantics is compositional.

Theorem 9 (Local Error Semantics for Composition) *For two composable EIOs S_1, S_2 and $S_{12} = S_1 \mid S_2$ we have:*

1. $ET_{12} = cont\Big(prune\Big(\big(ET_1 \mid EL_2\big) \cup \big(EL_1 \mid ET_2\big)\Big)\Big)$
2. $EL_{12} = \big(EL_1 \mid EL_2\big) \cup ET_{12}$

Proof. 1.a) '$\subseteq$':
Since both sides are closed under *cont*, it suffices to consider a prefix-minimal element w of ET_{12}. This means w is in MIT_{12} or in PrT_{12}.

First we consider the case, that $w \in MIT_{12}$:
We know that $w = xa$ with $(q_{01}, q_{02}) \stackrel{x}{\Rightarrow} (q_1, q_2) \stackrel{a}{\not\rightarrow}$, $a \in I_{12}$. Since $a \in I_{12}$, it holds that $a \in I_1 \dot\cup I_2$ and $a \notin O_1 \cup O_2$. Let w.l.o.g. $a \in I_1$. Thus by projection we get $q_{01} \stackrel{x_1}{\Rightarrow} q_1 \stackrel{a}{\not\rightarrow}$ and $q_{02} \stackrel{x_2}{\Rightarrow}$ (i.e. $x_2 \in L_2$) with $x \in x_1 \mid x_2$. Thus we know that $x_1 a \in ET_1$ and $x_2 \in L_2 \subseteq EL_2$, and it follows that $w \in (x_1 \mid x_2) \cdot \{a\} \subseteq x_1 a \mid x_2 \subseteq ET_1 \mid EL_2$, which is contained in the r.h.s. set.

Now we get to the second case: $w \in PrT_{12}$
In this case we know that there exists $u \in O_{12}^*$ such that $(q_{01}, q_{02}) \stackrel{w}{\Rightarrow} (q_1, q_2) \stackrel{u}{\Rightarrow} (q_1', q_2')$ with $(q_1', q_2') \in E_{12}$ and $w = prune(wu)$.

By projection we get $q_{01} \stackrel{w_1}{\Rightarrow} q_1 \stackrel{u_1}{\Rightarrow} q_1'$ and $q_{02} \stackrel{w_2}{\Rightarrow} q_2 \stackrel{u_2}{\Rightarrow} q_2'$ with $w \in w_1 \mid w_2$ and $u \in u_1 \mid u_2$. Since $(q_1', q_2') \in E_{12}$ it follows that either (q_1', q_2') is an inherited error, due to $q_1' \in E_1$ or $q_2' \in E_2$ or it is a new error due to some $a \in O_1 \cap I_2$ with $q_1' \stackrel{a}{\rightarrow} \wedge q_2' \stackrel{a}{\not\rightarrow}$ or some $a \in I_1 \cap O_2$ with $q_1' \stackrel{a}{\not\rightarrow} \wedge q_2' \stackrel{a}{\rightarrow}$.

If it is an inherited error, then let $q_1' \in E_1$ w.l.o.g. Thus we know that $w_1 u_1 \in ET_1$. Because of $q_{02} \stackrel{w_2 u_2}{\Rightarrow}$, we get $w_2 u_2 \in L_2 \subseteq EL_2$. Hence $wu \in ET_1 \mid EL_2$ and $w = prune(wu)$ is in the r.h.s. set.

If (q_1', q_2') is a new error, let w.l.o.g. $a \in I_1 \cap O_2$ with $q_1' \stackrel{a}{\not\rightarrow} \wedge q_2' \stackrel{a}{\rightarrow}$. Thus we know that $w_1 u_1 a \in MIT_1 \subseteq ET_1$ and $w_2 u_2 a \in L_2 \subseteq EL_2$. By definition of $\mid$ we know that $w_1 u_1 a \mid w_2 u_2 a = w_1 u_1 \mid w_2 u_2$ and thus we are done as above.

1.b) '$\supseteq$':
It should be noted that $S_1 \parallel S_2$ an $S_1 \mid S_2$ have the same states, error states and input actions. Consequently, using the *prune*-function on some trace of $S_1 \parallel S_2$ yields $v = \varepsilon$ or v ending on some $b \in I_{12} = I_{S_1 \mid S_2}$.

Again it suffices to consider a prefix-minimal element x. For such an x it holds that:
$x \in prune\Big(\big(ET_1 \mid EL_2\big) \cup \big(ET_2 \mid EL_1\big)\Big)$
Since x is the result of the *prune* function, we consider $xy \in \big(ET_1 \mid EL_2\big) \cup \big(ET_2 \mid EL_1\big)$ with $y \in O_{12}^*$. W.l.o.g we assume $xy \in ET_1 \mid EL_2$, i.e. there is $w_1 \in ET_1$ and $w_2 \in EL_2$ with $xy \in w_1 \mid w_2$. We also get $w \in w_1 \parallel w_2$ such that $w|_{\Sigma_{12}} = xy$.

Below, we will treat several cases, and in each case we will show that there is some $v \in PrT(S_1 \parallel S_2) \cup MIT(S_1 \parallel S_2)$ which is a prefix of w and either ends on an input action of $S_1 \mid S_2$ or is ε. In both cases $v|_{\Sigma_{12}}$ is a prefix of x. In case of $v|_{\Sigma_{12}} = \varepsilon$, the latter is obvious. Otherwise $v|_{\Sigma_{12}}$ ends on some input action $b \in I_{12}$ and it has to be a prefix of xy by construction of w. Since $y \in O_{12}^*$, this $v|_{\Sigma_{12}}$ has to be a prefix of x. Therefore x has a prefix in $PrT(S_1 \mid S_2) \cup MIT(S_1 \mid S_2)$ and we are done.

Let v_1 be the shortest prefix of w_1 that is in $PrT_1 \cup MIT_1$. If $w_2 \in L_2$, let $v_2 = w_2$; otherwise, let v_2 be the shortest prefix of w_2 that is in $PrT_2 \cup MIT_2$. Every action of v_1 and v_2 has its corresponding action in w. We now assume that $v_2 = w_2 \in L_2$ or the last action of v_1 is before or the same as the last action of v_2. Otherwise, $v_2 \in PrT_2 \cup MIT_2$ ends before v_1 and this is analogous to the case where v_1 ends before v_2. (Note that the case $v_2 = w_2 \in L_2$ is needed to cover the situation where w_2 ends before v_1, but is not an error trace).

If $v_1 = \varepsilon$, then choose $v_2' = v' = \varepsilon$.

If $v_1 \neq \varepsilon$, then v_1 by choice ends with some $a \in I_1$, i.e. $v_1 = v_1' a$. Let v' be the prefix of w that ends with the last action of v_1 and let $v_2' = v'|_{\Sigma_2}$. If $v_2 \in L_2 \cup PrT_2$, then v_2' is a prefix of v_2. If $v_2 \in MIT_2$ then it ends with some $b \in I_2$, i.e. $b \neq a$; according to the above assumption, in this case v_1 must end before v_2 and v_2' is a proper prefix of v_2.

In all cases (including the case $v_1 = \varepsilon$), we get (*) $q_{02} \overset{v_2'}{\Rightarrow}$. Furthermore, $v_2' = v'|_{\Sigma_2}$ is a prefix of v_2, and $v' \in v_1 \parallel v_2'$ is a prefix of w. Now we have to consider two cases:

First we consider the case, that $v_1 \in MIT_1$ (and $v_1 \neq \varepsilon$ in this case):
In this case we have $q_{01} \overset{v_1'}{\Rightarrow} q_1 \overset{a}{\not\rightarrow}$ and we let $v' = v''a$. We have to consider two subcases:

(i) If a is not a synchronizing action, i.e. $a \notin \Sigma_2$, then by (*) $q_{02} \overset{v_2'}{\Rightarrow} q_2$ with $v'' \in v_1' \parallel v_2'$. Therefore $(q_{01}, q_{02}) \overset{v''}{\Rightarrow} (q_1, q_2) \overset{a}{\not\rightarrow}$ with $a \in I_{12}$. Thus we can choose $v := v''a = v' \in MIT(S_1 \parallel S_2)$.

(ii) If $a \in \Sigma_2$, then $a \in O_2$ and $v_2' = v_2''a$. By (*) $q_{02} \overset{v_2''}{\Rightarrow} q_2 \overset{a}{\rightarrow}$ with $v'' \in v_1' \parallel v_2''$. Thus, $(q_{01}, q_{02}) \overset{v''}{\Rightarrow} (q_1, q_2)$ with $q_1 \overset{a}{\not\rightarrow}$, $a \in I_1$, $q_2 \overset{a}{\rightarrow}$ and $a \in O_2$; hence $(q_1, q_2) \in E_{12}$. In this case we choose $v := prune(v'') \in PrT(S_1 \parallel S_2)$.

The second case is $v_1 \in PrT_1$ (where we might have $v_1 = \varepsilon$).
In this case $\exists u_1 \in O_1^* : q_{01} \overset{v_1}{\Rightarrow} q_1 \overset{u_1}{\Rightarrow} q_1'$ with $q_1' \in E_1$.
Again $q_{02} \overset{v_2'}{\Rightarrow} q_2$, this time with $(q_{01}, q_{02}) \overset{v'}{\Rightarrow} (q_1, q_2)$. We have two subcases depending on 'how long' q_2 can 'take part' in u_1.

(i) There is some $u_2 \in (O_1 \cap I_2)^*$ and some $c \in (O_1 \cap I_2)$ such that u_2c is a prefix of $u_1|_{I_2}$ with $q_2 \overset{u_2}{\Rightarrow} q_2' \overset{c}{\not\rightarrow}$.
Consider the prefix $u_1'c$ of u_1 with $u_1'c|_{I_2} = u_2c$. We know that $q_1 \overset{u_1'}{\Rightarrow} q_1'' \overset{c}{\rightarrow}$.
Then $u_1' \in u_1' \parallel u_2$ and $(q_1, q_2) \overset{u_1'}{\Rightarrow} (q_1'', q_2') \in E_{12}$, i.e. we get a new error.
We can choose $v := prune(v'u_1') \in PrT(S_1 \parallel S_2)$, which is a prefix of v', since $u_1' \in O_1^*$.

(ii) Otherwise we have $q_2 \overset{u_2}{\Rightarrow} q_2'$ with $u_2 = u_1|_{I_2}$. Then $u_1 \in u_1 \parallel u_2$ and $(q_1, q_2) \overset{u_1}{\Rightarrow} (q_1', q_2') \in E_{12}$. This is an error inherited from S_1. Therefore we can again choose $v := prune(v'u_1) \in PrT(S_1 \parallel S_2)$, which again is a prefix of v'.

2. Observe that: $L_i \subseteq EL_i$ and $ET_i \subseteq EL_i$. For better readability, we start from the right hand side of the equation:

$$
\begin{aligned}
&(EL_1 \mid EL_2) \cup ET_{12} = \\
&\big((L_1 \cup ET_1) \mid (L_2 \cup ET_2)\big) \cup ET_{12} = \\
&\underbrace{(L_1 \mid ET_2)}_{\subseteq ET_{12} \quad (1)} \cup \underbrace{(ET_1 \mid L_2)}_{\subseteq ET_{12} \quad (1)} \cup (L_1 \mid L_2) \cup \underbrace{(ET_1 \mid ET_2)}_{\subseteq ET_{12} \quad (1)} \cup ET_{12} = \\
&(L_1 \mid L_2) \cup ET_{12} \overset{5.2}{=} L_{12} \cup ET_{12} = EL_{12}
\end{aligned}
$$

$\square$

The above theorem implies that $\sqsubseteq_{loc}$ is a precongruence. The main point is now to show that it is the coarsest one. To show this, we will construct a test environment U for each relevant trace w of $Impl$ that reveals that w is also a suitable trace of $Spec$.

Theorem 10 (Full Abstractness for Local Error Semantics) *For two systems Impl and Spec with the same signature it holds that:*
$Impl \sqsubseteq^c_{loc} Spec \Leftrightarrow Impl \sqsubseteq_{loc} Spec$

Proof. As just noted, it follows easily from Theorem 9 that $\sqsubseteq_{loc}$ is a precongruence. Furthermore, $\varepsilon \in ET(S)$ signifies that an error is locally reachable in S, since this can only result from $\varepsilon \in PrT(S)$. Hence, $Impl \sqsubseteq_{loc} Spec$ implies that $\varepsilon \in ET(Spec)$ whenever $\varepsilon \in ET(Impl)$, and thus also that $Impl \sqsubseteq^B_{loc} Spec$.

It remains to show that $Impl \sqsubseteq^c_{loc} Spec \Rightarrow Impl \sqsubseteq_{loc} Spec$. Since $Impl$ and $Spec$ have the same signature, we will write I for $I_{Impl} = I_{Spec}$ and O for $O_{Impl} = O_{Spec}$ throughout this proof.
We assume $Impl \sqsubseteq^c_{loc} Spec$, hence $Impl \sqsubseteq^B_{loc} Spec$ and $Impl \mid U \sqsubseteq^B_{loc} Spec \mid U$ for all EIOs U composable with $Impl$.
We have to show the following inclusions:

- $ET(Impl) \subseteq ET(Spec)$
- $EL(Impl) \subseteq EL(Spec)$

For the first inclusion we consider a prefix minimal element $w \in ET(Impl)$. It suffices to show that w or any of its prefixes is in $ET(Spec)$.
If $w = \varepsilon$, then an error state is locally reachable in $Impl$, hence also in $Spec$, because of $Impl \sqsubseteq^B_{loc} Spec$. Therefore $\varepsilon \in PrT(Spec)$.
So we assume that $w = x_1 \cdots x_n x_{n+1} \in \Sigma^+$ with $n \geq 0$ and $x_{n+1} \in I$. We consider the following process U(see Fig. 1):

- $Q_U = \{q_0, q_1, \ldots q_{n+1}\}$
- $I_U = O$
- $O_U = I$
- $q_{0U} = q_0$
- $E_U = \emptyset$
- $\delta_U = \{(q_i, x_{i+1}, q_{i+1}) \mid 0 \leq i \leq n\} \cup \{(q_i, x, q_{n+1}) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(q_{n+1}, a, q_{n+1}) \mid a \in I_U\}$

For w we can distinguish two cases. Both will lead to $\varepsilon \in StT(Impl \mid U)$.
If $w \in MIT(Impl)$, then in $Impl \parallel U$ we have $(q_{0Impl}, q_0) \stackrel{x_1 \cdots x_n}{\Rightarrow} (q', q_n)$ with $q' \stackrel{x_{n+1}}{\not\longrightarrow}$ and $q_n \stackrel{x_{n+1}}{\longrightarrow}$. Therefore $(q', q_n) \in E_{Impl|U}$ and $\varepsilon \in StT(Impl \mid U)$.
If $w \in PrT(Impl)$, then in $Impl \parallel U$ we have $(q_{0Impl}, q_0) \stackrel{w}{\Rightarrow} (q'', q_{n+1}) \stackrel{u}{\Rightarrow} (q', q_{n+1})$ with some $u \in O^*$ and $q' \in E_{Impl}$. The latter implies $(q', q_{n+1}) \in E_{Impl|U}$, and again $\varepsilon \in StT(Impl \mid U)$.
Since we now know that $\varepsilon \in StT(Impl \mid U)$, we also know from $Impl \mid U \sqsubseteq^B_{loc} Spec \mid U$ that there is a locally reachable error in $Spec \mid U$ as well.
There are two kinds of error states $Spec \mid U$ can have: new or inherited. Since

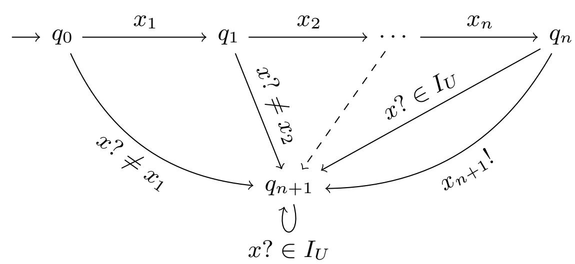


Fig. 1. $x? \neq x_i$ indicates all $x \in I_U \backslash \{x_i\}$, $x_{n+1}!$ indicates $x_{n+1} \in O_U$

each state of U enables every $x \in O = I_U$, a locally reachable new error has to be one where U enables an output $a \in O_U$ which is not enabled in *Spec*. By construction q_{n+1} enables no outputs, therefore such a new error state has to be of the form (q', q_i) with $i \leq n$, $q' \not\xrightarrow{x_i+1}$ and $x_{i+1} \in O_U = I$. Thus, by projection $q_{0Spec} \stackrel{x_1 \cdots x_i}{\Rightarrow} q' \not\xrightarrow{x_i+1}$ and therefore $x_1 \cdots x_{i+1} \in MIT(Spec) \subseteq ET(Spec)$ is a prefix of w and we are done.
If the locally reachable error is due to an inherited error state, then by projection U has performed some $x_1 \cdots x_i u$ with $u \in I_U^* = O^*$ (possibly $i = 0$) and hence so has *Spec*. With this, *Spec* has reached some state in E_{Spec}. Therefore $prune(x_1 \cdots x_i u) = prune(x_1 \cdots x_i) \in StT(Spec)$. Again this is a prefix of w and in $ET(Spec)$ and we are done.

For the second inclusion it suffices to show $L(Impl) \backslash ET(Impl) \subseteq EL(Spec)$, because of the first inclusion and the definition of EL.
For this we consider a $w \in L(Impl) \backslash ET(Impl)$ and show that it is in $EL(Spec)$. If $w = \varepsilon$ we are done, since ε always is in $EL(Spec)$. Therefore we consider $w = x_1 \cdots x_n$ with $n \geq 1$ and construct an EIO U (illustrated in Fig. 2) with:

- $Q_U = \{q, q_0, q_1, \ldots q_n\}$
- $I_U = O$
- $O_U = I$
- $q_{0U} = q_0$
- $E_U = \{q_n\}$
- $\delta_U = \{(q_i, x_{i+1}, q_{i+1}) \mid 0 \leq i < n\} \cup \{(q_i, x, q) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(q, x, q) \mid x \in I_U\}$

Because of $q_{0Impl} \stackrel{w}{\Rightarrow} q$ we know that $Impl \mid U$ has a locally reachable error. Thus, because of $Impl \mid U \sqsubseteq_{loc}^c Spec \mid U$, *Spec* also has to have a locally reachable error state. Firstly this could be a new error because of some $x_i \in O_U$ and $q_{0Spec} \stackrel{x_1 \cdots x_{i-1}}{\Rightarrow} q' \not\xrightarrow{x_i}$. In this case $x_1 \cdots x_i \in MIT(Spec)$ and thus $w \in EL(Spec)$. Note, that outputs of U are only enabled along this trace. Therefore there are no other outputs of U, which could lead to a new error.
Secondly it could be a new error due to some $a \in O_{Spec}$, which U could not

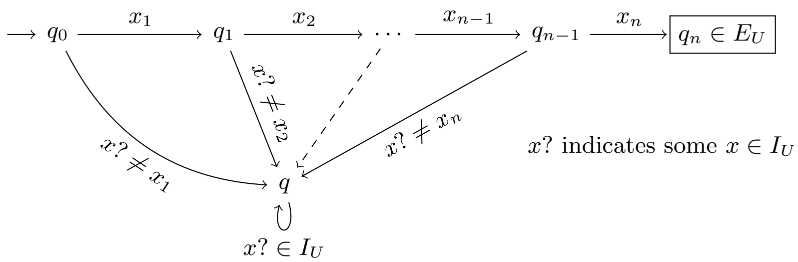


Fig. 2. Here and in the following error states are marked with a box.

match. But the only state of U in which not all inputs are enabled is q_n, which already is an error state. If this state is reachable in $Spec \mid U$, then the composed EIO has an inherited error and thus $w \in L(Spec) \subseteq EL(Spec)$
Thirdly it can be an error inherited from U. Since the only state in E_U is q_n and all actions are synchronized, this is only possible if $q_{0Spec} \overset{x_1 \cdots x_n}{\Rightarrow}$. In this case $w \in L(Spec)$ and we are done.
Finally, the error could have been inherited from $Spec$. In this case $q_{0Spec} \overset{x_1 \cdots x_i u}{\Rightarrow} q' \in E_{Spec}$, for some $i \geq 0$ and $u \in O^*$. This means that $x_1 \cdots x_i u \in StT(Spec)$ and thus $prune(x_1 \cdots x_i u) = prune(x_1 \cdots x_i) \in PrT(Spec) \subseteq EL(Spec)$. Hence again $w \in EL(Spec)$ and we are done. $\square$

3.2 Comparison to Interface Automata

Up to local-error equivalence, we can essentially work with EIOs without error states. Such EIOs are exactly the *interface automata* of [4], if they additionally are *input-deterministic*: if $q \xrightarrow{a} q'$ and $q \xrightarrow{a} q''$ for some $a \in I$, then $q' = q''$. The only difference is that with EIOs without error states we do not have EIOs anymore that, from the local point of view, are an error initially.

Theorem 11 (Removing Error States) *Let S be an EIO, and let $prune(S)$ be obtained from S by removing the* illegal *states in $illegal(S) = \{q \in Q \mid$ an error state is reachable from q by local actions$\}$, the respective transitions and all transitions $q \xrightarrow{a} q'$ where $a \in I$ and there is some $q \xrightarrow{a} q''$ with $q'' \in illegal(S)$. If $q_0 \notin illegal(S)$, $prune(S)$ is an EIO and local-error equivalent to S.*

Output pruning in [4] only removes the illegal states and the respective transitions. The additional removal of transitions $q \xrightarrow{a} q'$ as described in the theorem is obviously redundant in case of input determinism.

Proof. We assume $q_0 \notin illegal(S)$; then the claim about $prune(S)$ being an EIO is obvious.

For $S \sqsubseteq_{loc} prune(S)$, consider some $w \in PrT(S)$ and a suitable underlying run $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots q_n$. Observe that q_n is an illegal state and missing in $prune(S)$. So let q_i be the first state on the run such that $q_i \xrightarrow{a_{i+1}} q_{i+1}$ is missing in $prune(S)$. This means that q_i is not illegal, a_{i+1} is an input, and some q with $q_i \xrightarrow{a_{i+1}} q$ is illegal. This implies that some prefix of w is in $MIT(prune(S))$, and $w \in ET(prune(S))$.

For $wa \in MIT(S)$, we argue similarly. Either some suitable run underlying w is still in $prune(S)$ and $wa \in MIT(prune(S))$, or some transition of the run is missing in $prune(S)$ and wa has a prefix in $MIT(prune(S))$. Thus, $ET(S) \subseteq ET(prune(S))$.

Analogously for $w \in L(S)$, either some run underlying w is still in $prune(S)$ and $w \in L(prune(S))$, or some transition of the run is missing and w has a prefix in $MIT(prune(S))$. Thus, $EL(S) \subseteq EL(prune(S))$.

For $ET(prune(S)) \subseteq ET(S)$, we just have to consider $wa \in MIT(prune(S))$, where $a \in I$. Consider a suitable run $q_0 \dots q$ underlying w in $prune(S)$. Either, q has no a-transition in S and $wa \in MIT(S)$, or $q \xrightarrow{a} q'$ for some illegal q' and $wa \in PrT(S)$. Thus, $ET(prune(S)) \subseteq ET(S)$.

For $w \in L(prune(S))$, each run underlying w is still in S and $w \in L(prune(S))$. Thus, $EL(S) \subseteq EL(S)$. □

Thus, we could work with EIOs without error states; whenever we put such EIOs in parallel, we have to normalize the result, i.e. we take $prune(S_1 \mid S_2)$ as parallel composition. We only have to make sure that this is well-defined: we call EIOs S_1 and S_2 *compatible*, if the initial state of $S_1 \mid S_2$ is not illegal; with this, we only apply the new parallel composition to compatible S_1 and S_2. Furthermore, we have:

Proposition 12 *If Spec and Spec′ are compatible EIOs and Impl $\sqsubseteq_{loc}$ Spec, then also Impl and Spec′ are compatible.*

Proof. If $Impl$ and $Spec'$ are not compatible, then $\varepsilon \in ET(Impl \mid Spec')$. Now $ET(Impl \mid Spec') \subseteq ET(Impl \mid Spec)$ by Theorem 9, hence also $Impl$ and $Spec$ are not compatible. □

Thus, also on the level of transition systems, output pruning as introduced in [4] is justified according to our approach. But the refinement relation based on alternating simulation is somewhat arbitrarily too strict, as we will show below. To our best knowledge, alternating simulation as refinement relation has first been considered for modal transition systems [6]; see [7] for a comparison to the setting of interface automata.

Since the refinement relation of [4] is a precongruence, one might believe that, due to our coarsest precongruence result, this refinement should directly imply $\sqsubseteq_{loc}$. This is not really so obvious: we have considered parallel components that are not interface automata, and this could have forced us to be too strict w.r.t alternating simulation. But actually, this is not the case, as we show now.

Definition 13. For EIOs S_1 and S_2 with the same signature, an *alternating simulation relation* is some $\mathcal{R} \subseteq Q_1 \times Q_2$ with $(q_{01}, q_{02}) \in \mathcal{R}$ such that for all $(q_1, q_2) \in \mathcal{R}$ we have:

1. If $q_2 \xrightarrow{a} q_2'$ and $a \in I_1$, then $q_1 \xrightarrow{a} q_1'$ and $(q_1', q_2') \in \mathcal{R}$.
2. If $q_1 \xrightarrow{a} q_1'$ and $a \in O_1$, then $q_2 \Rightarrow\xrightarrow{a} q_2'$ and $(q_1', q_2') \in \mathcal{R}$.
3. If $q_1 \xrightarrow{\tau} q_1'$, then $q_2 \Rightarrow q_2'$ and $(q_1', q_2') \in \mathcal{R}$.

Thus, such an $\mathcal{R}$ requires that the implementation S_1 can match a prescribed input immediately, while an output or τ is allowed for S_1 if the specification can match them by using arbitrarily many internal steps.

Proposition 14 *If there exists some alternating simulation relation $\mathcal{R}$ for interface automata S_1 and S_2, then $S_1 \sqsubseteq_{loc} S_2$.*

Proof. Since interface automata do not have error states, we just have to consider $wa \in MIT(S_1)$ with $a \in I$ for $ET(S_1) \subseteq ET(S_2)$. Take a suitable run in S_1 underlying w, and build up a matching run in S_2 as follows. To start with, the initial states are related according to $\mathcal{R}$. Each output or internal transition in S_1 can be matched according to 13.2 or 13.3, reaching related states again. If the runs have reached $(q_1, q_2) \in \mathcal{R}$ so far and the next transition is $q_1 \xrightarrow{a} q_1'$ with $a \in I_1$, then either q_2 does not have an a-transition and a prefix of w is in $MIT(S_2)$, or $q_2 \xrightarrow{a} q_2'$ and $(q_1', q_2') \in \mathcal{R}$ due to input determinism. If the run in S_1 ends and we have reached $(q_1, q_2) \in \mathcal{R}$, then q_2 cannot have an a-transition due to 13.1 and $wa \in MIT(S_2)$.

The treatment of $w \in L(S_1)$ is analogous, except that we do not have to consider a missing action after w at the end. □

Now we come to the example announced above. The interface automata in Fig. 3 show two interface automata, which are local-error equivalent – in particular, they have no inputs: there are no error traces and the basic languages are the same. But there exists no alternating refinement relation from the first to the second, since whichever way the second interface automaton matches o, it will forbid one of o_1 or o_2 afterwards.

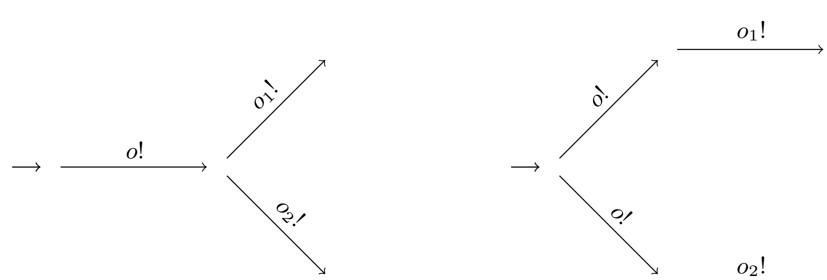


Fig. 3.

Finally, we show associativity for parallel composition. As mentioned in the introduction, Theorem 1 of [3] claiming this associativity is wrong due to an error in the definition of pruning; and such a proof is a bit tricky when composition involves pruning. The problem also demonstrates the danger when one develops an unorthodox definition justified with informal intuitive arguments only. In the present paper, pruning is proven correct in Theorem 11, and this proof would fail with some incorrect definition of pruning.

In our setting with error states, associativity is easy, because the two systems are easily seen to be isomorphic. Hence, the following result would also hold for any sensible equivalence on EIOs.

Theorem 15 *Let S_1, S_2 and S_3 EIOs. Then $S_1 \mid (S_2 \mid S_3)$ and $(S_1 \mid S_2) \mid S_3$ are local-error equivalent.*

Proof. Both EIOs are isomorphic to an EIO with state set $Q_1 \times Q_2 \times Q_3$, initial state (q_{01}, q_{02}, q_{03}), signature as that of $S_1 \mid (S_2 \mid S_3)$ and the following transitions and error states.

Transitions:

- $(q_1, q_2, q_3) \xrightarrow{\alpha} (q_1', q_2, q_3)$ if $q_1 \xrightarrow{\alpha} q_1'$ and $\alpha \in (\Sigma_1 \cup \{\tau\}) \backslash (\Sigma_2 \cup \Sigma_3)$ and similarly for transitions derived from S_2 and S_3 instead of S_1
  and
- $(q_1, q_2, q_3) \xrightarrow{\tau} (q_1', q_2', q_3)$ if $q_1 \xrightarrow{a} q_1'$ and $q_2 \xrightarrow{a} q_2'$ for some $a \in \Sigma_1 \cap \Sigma_2$ and similarly for transitions derived from other pairs instead of S_1 and S_2.

Error states: (q_1, q_2, q_3) is an error state

- if $q_1 \in E_1$ and similarly for E_2 or E_3 instead of E_1
  or
- if $q_1 \xrightarrow{a} q_1'$ and $\neg q_2 \xrightarrow{a}$ for some $a \in O_1 \cap I_2$ or similarly for one of the other five pairs instead of (S_1, S_2).

Observe that, in the latter item, (q_2, q_3) possibly is not an error state and $(q_1, (q_2, q_3)))$ is a new one, while (q_1, q_2) is an error state and $((q_1, q_2), q_3)$ is an inherited one. □

4 Internal Errors

Now we consider errors reached by internal actions only. This view is even more optimistic than our first one, since errors reachable by output actions are no longer considered dangerous. Nevertheless our result will show, that the resulting semantics is not much different from the local error semantics. We will annotate each error trace with a set of output actions; if a system with this trace synchronizes with another one on these output actions, an error state can be reached with this trace.

Our base relation is now defined as:

Definition 16 (Internal Basic Relation). An *error is internally reachable* in S, if $\varepsilon \in StT(S)$. We have $Impl \sqsubseteq_{int}^{B} Spec$, whenever an error is internally reachable in $Impl$ only if an error is internally reachable in $Spec$.

Again we denote the fully abstract precongruence with respect to $\sqsubseteq_{int}^{B}$ and $\mid$ by $\sqsubseteq_{int}^{c}$.

As mentioned above, we add a set of outputs to each error trace thereby getting an error pair. The intuition is that, once the actions of a pruned trace have been taken, the system in a composition might be prevented from reaching an error internally because of an output action that has been pruned, but has not been synchronized on.

Irgendwie ne Wiederholung – ueberarbeiten

Definition 17 (out function). Given an EIO S, we define $out : \Sigma^* \to \mathfrak{P}(O)$ such that X consists of all output actions in w.

Definition 18 (Error Pair). An *error pair* over a signature (I, O) is a pair $(w, X) \in \big((I \cup O)^* \times \mathfrak{P}(O)\big)$ with $out(w) \subseteq X$.

Given two composable EIOs S_1, S_2, we define for an error pair (w, X) over (I_1, O_1) and $v \in \Sigma_2^*$:
$(w, X) \parallel v = \{(z, Y) \mid z \in w \parallel v, Y = X \cup out(v)\}$
$(w, X) \mid v = \{(z|_{\Sigma_{12}}, Y \cap \Sigma_{12}) \mid (z, Y) \in (w, X) \parallel v\}$

It is easy to see that these sets consist of error pairs over the signatures of $S_1 \parallel S_2$ and of $S_{12} = S_1 \mid S_2$ resp. On error pairs over some (I, O), we define the following prefix relation and the following functions:
$(w, X) \sqsubseteq (v, Y)$ if $w \sqsubseteq v$ and $X \subseteq Y$
$prune(w, X) := (prune(w), X)$ (an error pair again)
$cont(w, X) := \{(v, Y) \mid (w, X) \sqsubseteq (v, Y)\}$ (consisting of error pairs)

Definition 19 (Sets of Error Pairs). We define the following sets of error pairs for an EIO S:

- strict error pairs: $StP(S) = \{(w, X) \mid w \in StT(S), out(w) = X\}$
- pruned error pairs: $PrP(S) = \{prune(w, X) \mid (w, X) \in StP(S)\}$
- missing input pairs: $MIP(S) = \{(w, X) \mid w \in MIT(S), out(w) = X\}$

It is easy to see that these sets indeed consist of error pairs over (I, O), and that they are an enhanced version of similar sets defined in the previous section. It will turn out that we can characterize $\sqsubseteq_{int}^c$ as componentwise set inclusion for pairs $(EP(S), EPL(S))$, where the latter is the basic language of S flooded with a set of traces derived from $EP(S)$; we introduce a sign for this relation.

Definition 20 (Internal Error Semantics). Let S be an EIO.

- The set of *error pairs* of S is $EP(S) = cont(PrP(S)) \cup cont(MIP(S))$;
- the set of *error pair traces* of S is $EPT(S) = \{w \mid (w, out(w)) \in EP(S)\}$;
- the flooded language of S, called *error pair language*, is $EPL(S) = L(S) \cup EPT(S)$.

For two EIOs $Impl$ and $Spec$ with the same signature, we write

$$Impl \sqsubseteq_{int} Spec \text{ if } EP(Impl) \subseteq EP(Spec) \text{ and } EPL(Impl) \subseteq EPL(Spec)$$

and we call $Impl$ and $Spec$ *internal-error equivalent* if $Impl \sqsubseteq_{int} Spec$ and $Spec \sqsubseteq_{int} Impl$.

The following lemma collects a number of useful observations for the next proof.

Lemma 21 *Given two composable EIOs S_1, S_2, $w_1 \in \Sigma_1^*$ and $w_2 \in \Sigma_2^*$, we have*

1. $w \in w_1 \mid w_2 \Rightarrow out(w) = out(w_1) \setminus I_2 \cup out(w_2) \setminus I_1$
2. $w \in w_1 \mid w_2 \Rightarrow (w, out(w)) \in (w_1, out(w_1)) \mid w_2$
3. $w \in w_1 \mid w_2 \Rightarrow wa \in w_1 a \mid w_2$ *if* $a \in \Sigma_1 \setminus \Sigma_2$

Given an EIO S, an error pair (w, X) over (I, O) and a set M of error pairs, we have

4. $prune(w, X) \sqsubseteq (w, X)$
5. $(w, X) \in M \Rightarrow (w, X) \in cont(prune(M))$
6. $StP(S) \subseteq EP(S)$

Proof. In particular, Item 6 follows from 5, which in turn follows from 4. □

For the characterization result, it is again crucial that the internal error semantics is compositional. Since we will give part of the proof on the level of $\|$, we note the following relationship.

Lemma 22 *Given two composable EIOs S_1, S_2, we have for $S_{12} = S_1 \mid S_2$*

1. $PrP(S_{12}) = \{(w|_{\Sigma_{12}}, X \cap \Sigma_{12}) \mid (w, X) \in PrP(S_1 \parallel S_2)\}$
2. $MIP(S_{12}) = \{(w|_{\Sigma_{12}}, X \cap \Sigma_{12}) \mid (w, X) \in MIP(S_1 \parallel S_2)\}$

Theorem 23 (Internal Error Semantics for Composition) *For two composable EIOs S_1, S_2 and $S_{12} = S_1 \mid S_2$ we have:*

1. $EP_{12} = cont\Big(prune\Big(\big(EP_1 \mid EPL_2\big) \cup \big(EPL_1 \mid EP_2\big)\Big)\Big)$
2. $EPL_{12} = \big(EPL_1 \mid EPL_2\big) \cup EPT_{12}$

Proof. 1.a) '$\subseteq$':
Since both sides are closed under *cont*, it suffices to consider a prefix-minimal element (w, X) of EP_{12}. This means (w, X) is in MIP_{12} or in PrP_{12}.

First, we consider the case $(w, X) \in MIP_{12}$.
We know that $X = out(w)$, $w = xa$, $a \in (I_1 \dot\cup I_2) \setminus (O_1 \dot\cup O_2)$ with $(q_{01}, q_{02}) \stackrel{x}{\Rightarrow} (q_1, q_2) \stackrel{a}{\not\to}$ in $S_1 \mid S_2$. Let w.l.o.g. $a \in I_1$. Thus by projection we get $q_{01} \stackrel{x_1}{\Rightarrow} q_1 \stackrel{a}{\not\to}$ and $q_{02} \stackrel{x_2}{\Rightarrow}$ with $x \in x_1 \mid x_2$. Thus we know that $x_1 a \in MIT_1$ and thus $(x_1 a, out(x_1 a)) \in MIP_1 \subseteq EP_1$. We also know that $x_2 \in L_2 \subseteq EPL_2$, and it follows that $(w, X) \in ((x_1 a, out(x_1 a)) \mid x_2) \subseteq EP_1 \mid EPL_2$ by Lemma 22.2. This set is contained in the r.h.s. of 23.1 by 22.5.

Now we consider $(w, X) \in PrP_{12}$.
In this case we know that there exists $u \in O_{12}^*$ such that $(q_{01}, q_{02}) \stackrel{w}{\Rightarrow} (q_1, q_2) \stackrel{u}{\Rightarrow} (q_1', q_2')$ in $S_1 \mid S_2$ with $(q_1', q_2') \in E_{12}$ and $out(wu) = X \wedge w = prune(wu)$.

By projection we get $q_{01} \overset{w_1}{\Rightarrow} q_1 \overset{u_1}{\Rightarrow} q_1'$ and $q_{02} \overset{w_2}{\Rightarrow} q_2 \overset{u_2}{\Rightarrow} q_2'$ with $w \in w_1 \mid w_2$ and $u \in u_1 \mid u_2$. Since $(q_1', q_2') \in E_{12}$ it follows that either (q_1', q_2') is an inherited error, due to $q_1' \in E_1$ or $q_2' \in E_2$, or it is a new error due to some $a \in O_1 \cap I_2$ with $q_1' \overset{a}{\rightarrow} \wedge q_2' \overset{a}{\not\rightarrow}$ or some $a \in I_1 \cap O_2$ with $q_1' \overset{a}{\not\rightarrow} \wedge q_2' \overset{a}{\rightarrow}$.

For the case that (q_1', q_2') is an inherited error, let $q_1' \in E_1$ w.l.o.g.
Thus we know that $w_1 u_1 \in StT_1$, $(w_1 u_1, out(w_1 u_1)) \in StP_1 \subseteq EP_1$ by Lemma 22.6 and $w_2 u_2 \in EPL_2$. By Lemma 22.2 we get $(wu, X) \in (w_1 u_1, out(w_1 u_1)) \mid w_2 u_2 \subseteq EP_1 \mid EPL_2$. Therefore $prune(wu, X) = (w, X) \in prune(EP_1 \mid EPL_2)$, which is in the r.h.s. set.

Alternatively, in case that (q_1', q_2') is a new error, let w.l.o.g. $a \in I_1 \cap O_2$ with $q_1' \overset{a}{\not\rightarrow} \wedge q_2' \overset{a}{\rightarrow}$.
Thus we know that $w_1 u_1 a \in MIT_1$, $(w_1 u_1 a, out(w_1 u_1 a)) \in MIP_1 \subseteq EP_1$ and $w_2 u_2 a \in L_2 \subseteq EPL_2$. By definition of $\mid$, we get $wu \in w_1 u_1 \mid w_2 u_2 = w_1 u_1 a \mid w_2 u_2 a$ and $(wu, X) \in EP_1 \mid EPL_2$ by Lemma 22.2. As above $prune(wu, X) = (w, X) \in prune(EP_1 \mid EPL_2)$ which is in the r.h.s. set.

1.b) '$\supseteq$':
Again it suffices to consider a prefix-minimal element (x, X) of the r.h.s., i.e. $(x, X) \in prune\big((EP_1 \mid EPL_2) \cup (EPL_1 \mid EP_2)\big)$ with $x \in \{\varepsilon\} \cup \Sigma_{12}^* \cdot I_{12}$.
Since x is the result of the *prune* function, we consider $(xy, X) \in (EP_1 \mid EPL_2) \cup (EPL_1 \mid EP_2)$ with $y \in O_{12}^*$. W.l.o.g we assume $(xy, X) \in EP_1 \mid EPL_2$, i.e. there is $(w_1, X_1) \in EP_1$ and $w_2 \in EPL_2$ with $(xy, X) \in (w_1, X_1) \mid w_2$. Furthermore, there is some $(w, Y) \in (w_1, X_1) \parallel w_2$ such that $(w|_{\Sigma_{12}}, Y \cap \Sigma_{12}) = (xy, X)$; note that $Y = X_1 \cup out(w_2)$ by definition and thus $out(w) = out(w_1) \cup out(w_2) \subseteq Y$.

We will show that there is a $(v, V) \in PrP(S_1 \parallel S_2) \cup MIP(S_1 \parallel S_2)$ with $(v, V) \sqsubseteq (w, Y)$ such that v ends on some $b \in I_{12}$ or is ε. In both cases, we have $(v|_{\Sigma_{12}}, V \cap \Sigma_{12}) \sqsubseteq (x, X)$: We have already argued in the corresponding part of the proof for Theorem 9 that $v|_{\Sigma_{12}} \sqsubseteq x$; furthermore, $V \cap \Sigma_{12} \subseteq Y \cap \Sigma_{12} = X$. Therefore (x, X) has a prefix in $PrP(S_1 \mid S_2) \cup MIP(S_1 \mid S_2)$ and we are done.

Let (v_1, V_1) be a minimal prefix of (w_1, X_1) in $PrP_1 \cup MIP_1$. If $w_2 \in L_2$, let $v_2 = w_2$; otherwise, let (v_2, V_2) be a minimal prefix of $(w_2, out(w_2))$ in $PrP_2 \cup MIP_2$. We now assume that $v_2 = w_2 \in L_2$ or v_1 ends in w before, or with the same action as, v_2; cf. the proof of Theorem 9. Otherwise, v_2 ends before v_1, and this is analogous to the case where v_1 ends before v_2: For this case, the proof below needs in particular $(v_1, V_1) \in PrP_1 \cup MIP_1$ and $V_1 \subseteq Y$, and if $v_2 \neq w_2$ ends before v_1 then we have symmetrically $(v_2, V_2) \in PrP_2 \cup MIP_2$ and $V_2 \subseteq Y$ due to $V_2 \subseteq out(w_2)$.

If $v_1 = \varepsilon$, then choose $v_2' = v' = \varepsilon$.

If $v_1 \neq \varepsilon$, then by choice v_1 ends with some $a \in I_1$ by definition of PrP and MIP, i.e. $v_1 = v_1' a$. Let v' be the prefix of w that ends with the last action of v_1 and let $v_2' = v'|_{\Sigma_2}$.
If $v_2 = w_2 \in L_2$ or $(v_2, V_2) \in PrP_2$, then v_2' is a prefix of v_2. If $(v_2, V_2) \in MIP_2$, then v_2 ends with some $b \in I_2$, i.e. $b \neq a$; thus, according to the above assumption, in this case v_1 must end before v_2 and v_2' is a proper prefix of v_2.

Either way (also for $v_1 = \varepsilon$) the following claims hold:

- $q_{02} \overset{v_2'}{\Rightarrow}$ (*)
- $v_2' = v'|_{\Sigma_2} \sqsubseteq v_2$
- $v' \in v_1 \parallel v_2'$ is prefix of w and thus $out(v') \subseteq out(w) \subseteq Y$

Now we have to consider two cases:

In the first case, $(v_1, V_1) \in MIP_1$ – and therefore $v_1 \neq \varepsilon$ and $V_1 = out(v_1)$. In this case we have $q_{01} \overset{v_1'}{\Rightarrow} q_1 \overset{a}{\not\rightarrow}$ and we let $v' = v''a$. We have to consider two subcases:

(i) If a is not a synchronizing action, i.e. $a \notin \Sigma_2$, then by (*) $q_{02} \overset{v_2'}{\Rightarrow} q_2$ with $v'' \in v_1' \parallel v_2'$. Therefore $(q_{01}, q_{02}) \overset{v''}{\Rightarrow} (q_1, q_2) \overset{a}{\not\rightarrow}$ with $a \in I_{12}$. Thus we can choose $(v, V) = (v''a, out(v')) \in MIP(S_1 \parallel S_2)$, since $v''a = v'$ is a prefix of w and $out(v') \subseteq Y$.

(ii) If $a \in \Sigma_2$, then $a \in O_2$ and $v_2' = v_2''a$. By (*) $q_{02} \overset{v_2''}{\Rightarrow} q_2 \overset{a}{\rightarrow}$ with $v'' \in v_1' \parallel v_2''$. Thus, $(q_{01}, q_{02}) \overset{v''}{\Rightarrow} (q_1, q_2)$ with $q_1 \overset{a}{\not\rightarrow}$, $a \in I_1$, $q_2 \overset{a}{\rightarrow}$ and $a \in O_2$; hence $(q_1, q_2) \in E_{12}$. In this case we choose $(v, V) := (prune(v''), out(v'')) \in PrP(S_1 \parallel S_2)$ and hence $v \sqsubseteq v'' \sqsubseteq v' \sqsubseteq w$ and $out(v'') \subseteq Y$.

The second case is $(v_1, V_1) \in PrP_1$.
In this case $\exists u_1 \in V_1^* \subseteq O_1^* : q_{01} \overset{v_1}{\Rightarrow} q_1 \overset{u_1}{\Rightarrow} q_1'$ with $q_1' \in E_1$; furthermore, $out(v_1 u_1) = V_1 \subseteq X_1 \subseteq Y$.
Again we have $q_{02} \overset{v_2'}{\Rightarrow} q_2$, and hence $(q_{01}, q_{02}) \overset{v'}{\Rightarrow} (q_1, q_2)$.
We have two subcases:

(i) q_2 cannot accept the sequence of inputs of S_2 contained in u_1, i.e. there is some $u_2 \in (O_1 \cap I_2)^*$ and some $c \in (O_1 \cap I_2)$ such that $u_2 c$ is a prefix of $u_1|_{I_2}$ with $q_2 \overset{u_2}{\Rightarrow} q_2' \overset{c}{\not\rightarrow}$.
Consider the prefix $u_1'c$ of u_1 with $u_1'c|_{I_2} = u_2 c$. We know that $q_1 \overset{u_1'}{\Rightarrow} q_1'' \overset{c}{\rightarrow}$.
Then $u_1' \in u_1' \parallel u_2$ and $(q_1, q_2) \overset{u_1'}{\Rightarrow} (q_1'', q_2') \in E_{12}$, i.e. we get a new error.
We can choose $(v, V) := (prune(v'u_1'), out(v'u_1')) \in PrP(S_1 \parallel S_2)$. Then $v \sqsubseteq v' \sqsubseteq w$ and $out(v'u_1') \subseteq out(w) \cup V_1 \subseteq Y$ and we are done.

(ii) Otherwise we have $q_2 \overset{u_2}{\Rightarrow} q_2'$ with $u_2 = u_1|_{I_2}$. Then $u_1 \in u_1 \parallel u_2$ and $(q_1, q_2) \overset{u_1}{\Rightarrow} (q_1', q_2') \in E_{12}$. This is an error inherited from S_1, since $q_1 \in E_1$. Similarly, we choose $(v, V) := (prune(v'u_1), out(v'u_1)) \in PrP(S_1 \parallel S_2)$, which again is a prefix of (w, Y).

2) First we check that (*) $EPT_1 \mid EPL_2 \subseteq EPT_{12}$ ($EPT_2 \mid EPL_1 \subseteq EPT_{12}$ is analogous):
Consider $w \in EPT_1 \mid EPL_2$. By projection $w \in w_1 \mid w_2 \wedge w_1 \in EPT_1 \wedge w_2 \in EPL_2$. Therefore $(w_1, out(w_1)) \in EP_1$. Because of this and Lemma 23.2 we get that $(w, out(w)) \in (w_1, out(w_1)) \mid w_2 \subseteq EP_1 \mid EPL_2 \subseteq EP_{12}$. Thus $w \in EPT_{12}$

by definition of EPT.
Now we prove the second item from left to right (for better readability). For the indicated inclusions, we need (*) and $L_1 \subseteq EPL_1$, $L_2 \subseteq EPL_2$ and $EPT_1 \subseteq EPL_1$.

$$
\begin{aligned}
&(EPL_1 \mid EPL_2) \cup EPT_{12} \\
&= \big((L_1 \cup EPT_1) \mid (L_2 \cup EPT_2)\big) \cup EPT_{12} \\
&= (L_1 \mid L_2) \cup \underbrace{(L_1 \mid EPT_2)}_{\subseteq EPT_{12}} \cup \underbrace{(EPT_1 \mid L_2)}_{\subseteq EPT_{12}} \cup \underbrace{(EPT_1 \mid EPT_2)}_{\subseteq EPT_{12}} \cup EPT_{12} \\
&= (L_1 \mid L_2) \cup EPT_{12} = L_{12} \cup EPT_{12} = EPL_{12} \square
\end{aligned}
$$

The above theorem implies that $\sqsubseteq_{int}$ is a precongruence. Again the main point is now to show that it is the coarsest one. To show this, we will construct a test environment U for each relevant trace or error pair of *Impl* that reveals that it is also a suitable trace or error pair of *Spec*. We cannot use the same environment we used for Theorem 10, as can be seen in this counterexample:
$I(Impl) = I(Spec) = O(U) = \{a\}$ and $O(Impl) = O(Spec) = I(U) = \{b\}$.

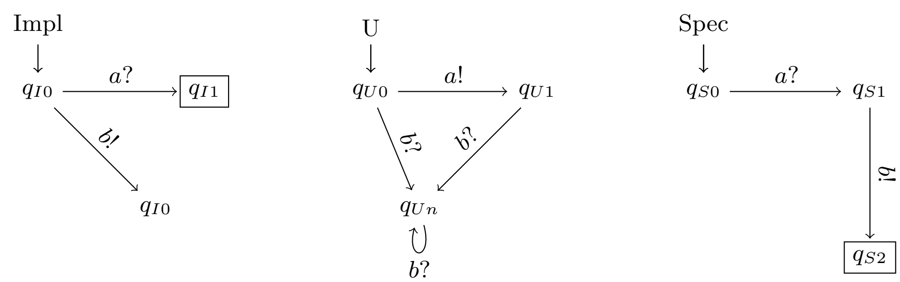


Fig. 4.

$(a?, \emptyset) \in EP(Impl)$, but even though $Impl \mid U$ and $Spec \mid U$ both have an error reachable by internal actions alone, $(a?, \emptyset) \notin Spec$.

Theorem 24 (Full Abstractness for Internal Error Semantics) *For two systems Impl and Spec with the same signature it holds that:*
$Impl \sqsubseteq^c_{int} Spec \Leftrightarrow Impl \sqsubseteq_{int} Spec$

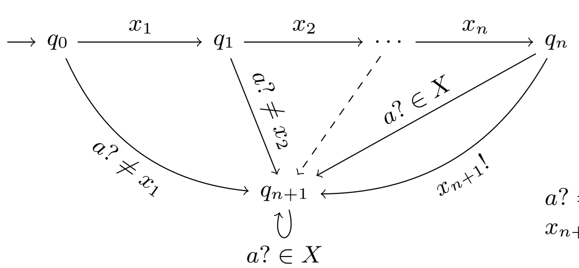
Proof. As just noted, it follows easily from Theorem 23 that $\sqsubseteq_{int}$ is a precongruence. Furthermore, $(\varepsilon, \emptyset) \in EP(S)$ signifies that an error is internally reachable in S, since this can only result from $(\varepsilon, \emptyset) \in PrP(S)$, which can only be if $\varepsilon \in StT(S)$. Hence, $Impl \sqsubseteq_{int} Spec$ implies that $(\varepsilon, \emptyset) \in EP(Spec)$ whenever $(\varepsilon, \emptyset) \in EP(Impl)$, and thus also that $Impl \sqsubseteq^B_{int} Spec$.

It remains to show that $Impl \sqsubseteq^c_{int} Spec \Rightarrow Impl \sqsubseteq_{int} Spec$. Since $Impl$ and $Spec$ have the same signature, we will write I for $I_{Impl} = I_{Spec}$ and O for $O_{Impl} = O_{Spec}$ throughout this proof.
We assume $Impl \sqsubseteq^c_{int} Spec$, hence $Impl \sqsubseteq^B_{int} Spec$ and $Impl \mid U \sqsubseteq^B_{int} Spec \mid U$ for all EIOs U composable with $Impl$.
We have to show the following inclusions:

- $EP(Impl) \subseteq EP(Spec)$
- $EPL(Impl) \subseteq EPL(Spec)$

For the first inclusion we consider a prefix-minimal element $(w, X) \in EP(Impl)$. It suffices to show that (w, X) or any of its prefixes is in $EP(Spec)$.
We first consider the case $w \neq \varepsilon$. So for $w = x_1 \cdots x_n x_{n+1} \in \Sigma^+$ with $n \geq 0$ and $x_{n+1} \in I$ we define the following process U(see Fig. 5):

- $Q_U = \{q_0, q_1, \dots q_{n+1}\}$
- $I_U = X$
- $O_U = I$
- $q_{0U} = q_0$
- $E_U = \emptyset$
- $\delta_U = \{(q_i, x_{i+1}, q_{i+1}) \mid 0 \leq i \leq n\} \cup \{(q_i, a, q_{n+1}) \mid a \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(q_{n+1}, a, q_{n+1}) \mid a \in I_U\}$



$a? \neq x_i$ indicates all $a \in X \backslash \{x_i\}$
$x_{n+1}!$ indicates $x_{n+1} \in O_U$

Fig. 5.

Note that $a, x_i \in (I \cup X)^*$ holds for all a and x_i and thus, by construction, they are hidden in the parallel composition.

First we show, that $(\varepsilon, \emptyset) \in EP(Impl \mid U)$, i.e. that $Impl \mid U$ has an internally reachable error. Because of $(w, X) \in EP(Impl)$ we can distinguish two cases, both resulting in $(\varepsilon, \emptyset) \in EP(Impl \mid U)$:
If $(w, X) \in MIP(Impl)$ then we have $(q_{0Impl}, q_0) \overset{x_1 \cdots x_n}{\Rightarrow} (q', q_n)$ in $Impl \parallel U$ and thus $(q_{0Impl}, q_0) \overset{\varepsilon}{\Rightarrow} (q', q_n)$ in $Impl \mid U$. We also have $q' \overset{x_{n+1}}{\not\longrightarrow}$ and $q_n \overset{x_{n+1}}{\longrightarrow}$. Therefore $(q', q_n) \in E_{Impl|U}$ and $(\varepsilon, \emptyset) \in StP(Impl \mid U)$.
If $(w, X) \in PrP(Impl)$, then we have in $Impl$: $q_{0Impl} \overset{w}{\Rightarrow} q'' \overset{u}{\Rightarrow} q' \in E_{Impl}$ with $u \in X^*$ by definition of EP. Thus in $Impl \parallel U$ we get: $(q_{0Impl}, q_0) \overset{w}{\Rightarrow}$

$(q'', q_{n+1}) \overset{u}{\Rightarrow} (q', q_{n+1}) \in E_{Impl \| U}$. Since all actions of w and u are in $I \cup X \subseteq Synch(Impl, U)$ we get: $(q_{0Impl}, q_0) \overset{\varepsilon}{\Rightarrow} (q'', q_{n+1}) \overset{\tau^{|u|}}{\Rightarrow} (q', q_{n+1}) \in E_{Impl|U}$ in $Impl \mid U$. Thus we get $(\varepsilon, \emptyset) \in StP(Impl \mid U)$.
Thus $Impl \mid U$ has an internal error and because of $Impl \mid U \sqsubseteq^B_{int} Spec \mid U$, $Spec \mid U$ must also have one, i.e. $(\varepsilon, \emptyset) \in EP(Spec \mid U)$. This error must be due to an error state, which is either new or inherited.
Since each state of U enables every $x \in X = I_U$ and all synchronized outputs of $Spec$ are in X, an internally reachable new error has to be one where U enables an output $x \in O_U$ which is currently not enabled in $Spec$. By construction q_{n+1} enables no outputs, therefore such a new error state has to be of the form (q', q_i) with $i \leq n$, $q' \overset{x_{i+1}}{\not\longrightarrow}$ and $x_{i+1} \in O_U = I$. Thus, by projection $q_{0Spec} \overset{x_1 \cdots x_i}{\Rightarrow} q' \overset{x_{i+1}}{\not\longrightarrow}$ and therefore $(x_1 \cdots x_{i+1}, out(x_1 \cdots x_{i+1}) \in MIP(Spec)$. Since $(x_1 \cdots x_{i+1}, out(x_1 \cdots x_{i+1}) \sqsubseteq (w, X)$ we get $(w, X) \in MIP(Spec) \subseteq EP(Spec)$ and are done.
If the internally reachable error is due to an inherited error state, then by projection U has performed some $x_1 \cdots x_i u$ with $u \in I_U^* = X^*$ and hence so has $Spec$. With this, $Spec$ has reached some state in E_{Spec}. Therefore
$prune((x_1 \cdots x_i u, out(x_1 \cdots x_i u))) = (prune(x_1 \cdots x_i u), out(x_1 \cdots x_i u)) =$
$(prune(x_1 \cdots x_i), out(x_1 \cdots x_i u)) \in StP(Spec)$. Again this is a prefix of (w, X) which is therefore in $EP(Spec)$ and we are done.

For $w = \varepsilon$, i.e. $(w, X) = (\varepsilon, X)$ we choose U as follows:

- $Q_U = \{q_0\}$
- $I_U = X$
- $O_U = I$
- $q_{0U} = q_0$
- $E_U = \emptyset$
- $\delta_U = \{(q_0, x, q_0) \mid x \in I_U\}$

Again we first show, that $(\varepsilon, \emptyset) \in EP(Impl \mid U)$. We know that $(\varepsilon, X) \notin MIP(Impl)$, since ε does not end in an input action. Therefore $(\varepsilon, X) \in PrP(Impl)$. Thus we have $q_{0Impl} \overset{u}{\Rightarrow} q' \in E_{Impl}$ with $u \in X^*$ as above. Analogously we get: $(q_{0Impl}, q_0)(q'', q_0) \overset{\varepsilon}{\Rightarrow} (q', q_{n+1}) \in E_{Impl|U}$ in $Impl \mid U$, and $(\varepsilon, \emptyset) \in StP(Impl \mid U)$.

Since $Impl \mid U$ has an internal error, $Spec \mid U$ must also have one. Since the state of U enables every $x \in X = I_U$ and all synchronized outputs of $Spec$ are in X, and since U has no outputs whatsoever, this error canonly be due to an inherited error state.

Thus by projection, U and $Spec$ can perform some $u \in I_U^* = X^*$, with $Spec$ reching an error state. Therefore $prune((u, out(u))) = (prune(u), out(u)) = (\varepsilon, out(u)) \in StP(Spec)$. Again this is a prefix of (w, X) which is therefore in $EP(Spec)$ and we are done.

For the second inclusion it suffices to show that $L(Impl) \setminus EPT(Impl) \subseteq EPL(Spec)$, since $EPT(Impl) \subseteq EPT(Spec)$ follows from the first inclusion.

For this we consider a $w \in L(Impl) \backslash EPT(Impl)$ with $w = x_1 \cdots x_n$. For $n = 0$ we are done, since $\varepsilon \in L(Spec)$ always holds. Note that $q_{0Impl} \stackrel{w}{\Rightarrow} q'$ and $\nexists w' \sqsubseteq w : q_{0Impl} \stackrel{w'}{\Rightarrow} q'' \in E_{Impl}$.

Now consider U with:

- $Q_U = \{q, q_0, q_1, \ldots q_n\}$
- $I_U = out(w)$
- $O_U = I_{Impl}$
- $q_{0U} = q_0$
- $E_U = \{q_n\}$
- $\delta_U = \{(q_i, x_{i+1}, q_{i+1}) \mid 0 \leq i < n\} \cup \{(q_i, x, q) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(q, x, q) \mid x \in I_U\}$

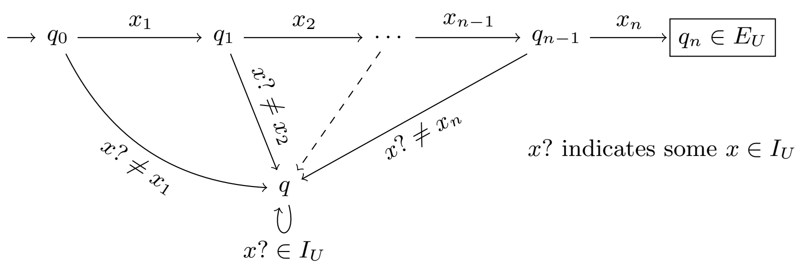


Fig. 6.

Because of $q_{0Impl} \stackrel{w}{\Rightarrow} q$ we know that $Impl \mid U$ has an internally reachable error. Thus, because of $Impl \mid U \sqsubseteq^c_{int} Spec \mid U$, $Spec$ also has to have a internally reachable error state. Firstly this could be a new error because of some $x_i \in O_U$ and $q_{0Spec} \stackrel{x_1 \cdots x_{i-1}}{\Rightarrow} q' \stackrel{x_i}{\not\longrightarrow}$. In this case $x_1 \cdots x_i \in MIT(Spec)$ and therefore $(x_1 \cdots x_i, out(x_1 \cdots x_i)) \in MIP(Spec)$. Since $(x_1 \cdots x_i, out(x_1 \cdots x_i)) \sqsubseteq (w, out(w))$, we get $(w, out(w)) \in EP(Spec)$ and thus $w \in EPT(Spec) \subseteq EPL(Spec)$. Note, that outputs of U are only enabled along this trace. Therefore there are no other outputs of U, which could lead to a new error.
Secondly it could be a new error due to some $a \in I_U$, which U could not match. But the only state of U in which not all inputs are enabled is q_n, which already is an error state. Therefore this would be an inherited error, which is described next.
Thirdly it can be an error inherited from U. Since the only state in E_U is q_n, this is only possible if $q_{0Spec} \stackrel{x_1 \cdots x_n}{\Rightarrow}$. In this case $w \in L(Spec)$ and we are done.
Finally, the error could have been inherited from $Spec$. In this case we have $q_{0Spec} \stackrel{x_1 \cdots x_i u}{\Rightarrow} q' \in E_{Spec}$, for some $i \geq 0$ and $u \in out(w)^*$ (no other outputs of $Spec$ are synchronized and consequently hidden). This means that $(x_1 \cdots x_i u, out(x_1 \cdots x_i u)) \in StP(Spec)$ and therefore $prune((x_1 \cdots x_i u, out(x_1 \cdots x_i u))) = (prune(x_1 \cdots x_i), out(x_1 \cdots x_i u)) \in PrP(Spec) \subseteq EP(Spec)$. Because of $u \in$

$out(w)^*$ we get $out(x_1 \cdots x_i u) \subseteq out(w)$. Together with $x_1 \cdots x_i \sqsubseteq w$ we get $(w, out(w)) \in EP(Spec)$ and therefore $w \in EPT(Spec) \subseteq EPL(w)$.

References

[1] Sebastian S. Bauer, Philip Mayer, Andreas Schroeder, and Rolf Hennicker. On weak modal compatibility, refinement, and the mio workbench. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2010*, LNCS 6015, pages 175–189. Springer, 2010.

[2] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[3] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE 2001*, pages 109–120, 2001.

[4] Luca de Alfaro and Thomas A. Henzinger. Interface-based design. In M. et al. Broy, editor, *Engineering Theories of Software Intensive Systems*, volume 195 of *NATO Science Series*, pages 1–148. Springer, 2005.

[5] D. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent circuits*. MIT Press, Cambridge, 1989.

[6] Kim Guldstrand Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, LNCS 407, pages 232–246. Springer, 1990.

[7] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal i/o automata for interface and product line theories. In R. de Nicola, editor, *ESOP*, LNCS 4421, pages 64–79. Springer, 2007.

[8] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.