1 Definitionen

Stand: 2. August 2017

Das Definitions-Kapitel wurde auf Grundlage von [BV15b] und [Sch16] zusammengestellt und teilweise von [BFLV16] beeinflusst.

Definition 1.1 (*Modal Error-I/O-Transitionssystem*). *Ein* Modal Error-I/O-Transitionssystem (MEIO) *ist ein Tupel* $(P, I, O, \longrightarrow, \dashrightarrow, p_0, E)$ *mit:*

- P*: Menge der* Zustände*,*
- $p_0 \in P$*:* Startzustand*,*
- I, O*: disjunkte Mengen der (sichtbaren)* Input- *und* Output-Aktionen*,*
- $\longrightarrow \subseteq P \times \Sigma_\tau \times P$*:* must-Transitions-Relation*,*
- $\dashrightarrow \subseteq P \times \Sigma_\tau \times P$*:* may-Transitions-Relation*,*
- $E \subseteq P$*: Menge der* Fehler-Zustände*.*

Es wird vorausgesetzt, dass $\longrightarrow \subseteq \dashrightarrow$ *(*syntaktische Konsistenz*) gilt.*
Das Alphabet *bzw. die Aktionsmenge eines MEIO ist* $\Sigma = I \cup O$*. Die* interne Aktion τ *ist nicht in* Σ *enthalten. Jedoch wird* $\Sigma_\tau := \Sigma \cup \{\tau\}$ *definiert. Die* Signatur *eines MEIOs entspricht* $\mathrm{Sig}(P) = (I, O)$*.*
Falls $\longrightarrow = \dashrightarrow$ *gilt, wird* P *auch* Implementierung *genannt.*

Implementierungen entsprechen den z.B. in [Sch16] behandelten EIOs.
Must-Transitionen sind Transitionen, die von einer Verfeinerung implementiert werden müssen. Die may-Transitionen sind hingegen die zulässigen Transitionen für eine Verfeinerung. Alle nicht vorhandenen Transitionen dürfen auch in keiner Verfeinerung einer Spezifikation in MEIO-Form auftauchen.
MEIOs werden in dieser Arbeit durch ihre Zustandsmenge (z.B. P) identifiziert und falls notwendig werden damit auch die Komponenten indiziert (z.B. I_P anstatt I). Falls das MEIO selbst bereits einen Index hat (z.B. P_1) kann an der Komponente die Zustandsmenge als Index wegfallen und nur noch der Index des gesamten Transitionssystems verwendet werden (z.B. I_1 anstatt I_{P_1}). Zusätzlich stehen i, o, a, ω und α für Buchstaben aus den Alphabeten $I, O, \Sigma, O \cup \{\tau\}$ und Σ_τ.
Es wird die Notation $p \overset{\alpha}{\dashrightarrow} p'$ für $(p, \alpha, p') \in \dashrightarrow$ und $p \overset{\alpha}{\dashrightarrow}$ für $\exists p' : (p, \alpha, p') \in \dashrightarrow$ verwendet. Dies kann entsprechend auf Buchstaben-Sequenzen $w \in \Sigma_\tau^*$ erweitert werden:

für $p \overset{w}{\dashrightarrow} p'$ ($p \overset{w}{\dashrightarrow}$) steht für die Existenz eines Laufes $p \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \dots p_{n-1} \overset{\alpha_n}{\dashrightarrow} p'$ ($p \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \dots p_{n-1} \overset{\alpha_n}{\dashrightarrow}$) mit $w = \alpha_1 \dots \alpha_n$.
Desweiteren soll $w|_B$ die Aktions-Sequenz bezeichnen, die man erhält, wenn man aus w alle Aktionen löscht, die nicht in $B \subseteq \Sigma$ enthalten sind. $\widehat{w}$ steht für $w|_\Sigma$. Es wird $p \overset{w}{\Longrightarrow} p'$ für ein $w \in \Sigma^*$ geschrieben, falls $\exists w' \in \Sigma_\tau^* : \widehat{w'} = w \wedge p \overset{w'}{\dashrightarrow} p'$, und $p \overset{w}{\Longrightarrow}$, falls $p \overset{w}{\Longrightarrow} p'$ für ein beliebiges p' gilt. Falls $p_0 \overset{w}{\Longrightarrow} p$ gilt, dann wird w *Trace* genannt und p ist ein *erreichbarer Zustand*.
Analog zu $\dashrightarrow$ und $\Longrightarrow$ werden $\longrightarrow$ und $\Longrightarrow$ für die entsprechenden Relationen der must-Transition verwendet.
Outputs und die interne Aktion werden *lokale Aktionen* genannt, da sie lokal vom ausführenden MEIO kontrolliert sind.
Um eine Erleichterung der Notation zu erhalten, soll gelten, dass $p \overset{a}{\not\longrightarrow}$ und $p \overset{a}{\not\dashrightarrow}$ für $\neg\exists p' : p \overset{a}{\longrightarrow} p'$ und $\neg\exists p' : p \overset{a}{\dashrightarrow} p'$ stehen soll. $p \overset{a}{\longrightarrow}\overset{\varepsilon}{\Longrightarrow} p'$ wird geschrieben, wenn p'' existiert, so dass $p \overset{a}{\longrightarrow} p'' \overset{\varepsilon}{\Longrightarrow} p'$ gilt. Diese Transition wird auch als *schwach-nachlaufende must-Transition* bezeichnet. Entsprechend steht $\overset{a}{\dashrightarrow}\overset{\varepsilon}{\Longrightarrow}$ für die *schwach-nachlaufende may-Transition*.
In Graphiken wird eine Aktion a als $a?$ notiert, falls $a \in I$ und $a!$, falls $a \in O$. Must-Transitionen (may-Transitionen) werden als durchgezogener Pfeil gezeichnet (gestrichelter Pfeil). Entsprechend der syntaktischen Konsistenz repräsentiert jede gezeichnete must-Transition auch gleichzeitig die zugrundeliegende may-Transitionen.

Definition 1.2 (*Parallelkomposition*). *Zwei MEIOs* $P_1 = (P_1, I_1, O_1, \longrightarrow_1, \dashrightarrow_1, p_{01}, E_1)$ *und* $P_2 = (P_2, I_2, O_2, \longrightarrow_2, \dashrightarrow_2, p_{02}, E_2)$ *sind* komponierbar*, falls* $O_1 \cap O_2 = \emptyset$. *Für solche MEIOs ist die* Parallelkomposition $P_{12} := P_1 \| P_2 = ((P_1 \times P_2), I, O, \longrightarrow_{12}, \dashrightarrow_{12}, (p_{01}, p_{02}), E)$ *definiert mit:* TODO: erzwungenen Zeilenumbrüche kontrollieren

- $I = (I_1 \cup I_2) \backslash (O_1 \cup O_2)$,
- $O = (O_1 \cup O_2)$,
- $$\begin{aligned} \longrightarrow_{12} = &\left\{((p_1,p_2),\alpha,(p_1',p_2)) \mid p_1 \overset{\alpha}{\longrightarrow}_1 p_1', \alpha \in \Sigma_\tau \backslash \mathrm{Synch}(P_1,P_2)\right\} \\ &\cup \left\{((p_1,p_2),\alpha,(p_1,p_2')) \mid p_2 \overset{\alpha}{\longrightarrow}_2 p_2', \alpha \in \Sigma_\tau \backslash \mathrm{Synch}(P_1,P_2)\right\} \\ &\cup \left\{((p_1,p_2),\alpha,(p_1',p_2')) \mid p_1 \overset{\alpha}{\longrightarrow}_1 p_1', p_2 \overset{\alpha}{\longrightarrow}_2 p_2', \alpha \in \mathrm{Synch}(P_1,P_2)\right\}, \end{aligned}$$
- $$\begin{aligned} \dashrightarrow_{12} = &\left\{((p_1,p_2),\alpha,(p_1',p_2)) \mid p_1 \overset{\alpha}{\dashrightarrow}_1 p_1', \alpha \in \Sigma_\tau \backslash \mathrm{Synch}(P_1,P_2)\right\} \\ &\cup \left\{((p_1,p_2),\alpha,(p_1,p_2')) \mid p_2 \overset{\alpha}{\dashrightarrow}_2 p_2', \alpha \in \Sigma_\tau \backslash \mathrm{Synch}(P_1,P_2)\right\} \\ &\cup \left\{((p_1,p_2),\alpha,(p_1',p_2')) \mid p_1 \overset{\alpha}{\dashrightarrow}_1 p_1', p_2 \overset{\alpha}{\dashrightarrow}_2 p_2', \alpha \in \mathrm{Synch}(P_1,P_2)\right\}, \end{aligned}$$
- $$\left.\begin{aligned} E = &(P_1 \times E_2) \cup (E_1 \times P_2) \\ &\cup \left\{(p_1,p_2) \mid \exists a \in O_1 \cap I_2 : p_1 \overset{a}{\dashrightarrow}_1 \wedge p_2 \overset{a}{\not\longrightarrow}_2\right\} \\ &\cup \left\{(p_1,p_2) \mid \exists a \in I_1 \cap O_2 : p_1 \overset{a}{\not\longrightarrow}_1 \wedge p_2 \overset{a}{\dashrightarrow}_2\right\} \end{aligned}\right\} \begin{aligned} &\textit{geerbte Fehler} \\ &\textit{neue Kommunikationsfehler.} \end{aligned}$$

Dabei bezeichnet $\text{Synch}(P_1, P_2) = (I_1 \cap O_2) \cup (O_1 \cap I_2) \cup (I_1 \cap I_2)$ *die Menge der zu* synchronisierenden Aktionen*. Die synchronisierten Aktionen werden als Inputs, in dem Fall* $(I_1 \cap I_2)$*, bzw. Outputs, in allen anderen Fällen, der Komposition beibehalten.*
P_1 *wird* Partner *von* P_2 *genannt, wenn die Parallelkomposition von* P_1 *und* P_2 *geschlossen ist. Eine Parallelkomposition von zwei MEIOs* P_1 *und* P_2 *ist* geschlossen*, wenn* P_1 *und* P_2 *duale Signaturen* $\text{Sig}(P_1) = (I, O)$ *und* $\text{Sig}(P_2) = (O, I)$ *haben.*

Ein neuer Fehler entsteht, wenn eines der MEIOs die Möglichkeit für einen Output hat (may-Transition) und das andere MEIO den passenden Input nicht sicher stellt (keine must-Transition vorhanden). Es muss also in möglichen Implementierungen nicht wirklich zu diesem Fehler kommen, da die Output-Transition nicht zwingendermaßen implementiert werden muss und die may-Input-Transition trotzdem erlaubt sein kann.
Wie bereits unter anderem in [Sch16] gesehen werden konnte, kann es durch die Synchronisation von Inputs zu keinen neuen Fehler kommen, da die Inputs in beiden Transitionssystemen keine lokal kontrollierten Aktionen sind. Falls jedoch nur eines der Transitionssysteme die Möglichkeit für einen Input hat, der synchronisiert wird, besteht diese Möglichkeit in der Parallelkomposition nicht mehr. Es kann also in der Kommunikation mit einem weiteren MEIO dort zu einem neuen Fehler kommen.

Definition 1.3 (*alternierende Simulation*). *Eine* (starke) alternierende Simulation *ist eine Relation* $\mathcal{R} \subseteq P \times Q$ *auf zwei MEIOs* P *und* Q*, wenn für alle* $(p, q) \in \mathcal{R}$ *mit* $q \notin E_Q$ *und* $\alpha \in \Sigma_\tau$ *gilt:*

1. $p \notin E_P$*,*
2. $q \xrightarrow{\alpha}_Q q'$ *impliziert* $p \xrightarrow{\alpha}_P p'$ *für ein* p' *mit* $p' \mathcal{R} q'$*,*
3. $p \overset{\alpha}{\dashrightarrow}_P p'$ *impliziert* $q \overset{\alpha}{\dashrightarrow}_Q q'$ *für ein* q' *mit* $p' \mathcal{R} q'$*.*

Die Vereinigung $\sqsubseteq_{\text{as}}$ *aller dieser Relationen wird als* (starke) as-Verfeinerung(-s Relation) *(auch modal Verfeinerung) bezeichnet. Es wird* $P \sqsubseteq_{\text{as}} Q$ *geschrieben, falls* $p_0 \sqsubseteq_{\text{as}} q_0$ *gilt.* $P \sqsubseteq_{\text{as}} Q$ *steht dabei dafür, dass* P Q (stark) as-verfeinert *oder dass* P *eine* (starke) as-Verfeinerung *von* Q *ist.*
Für ein MEIO Q *und eine Implementierung* P *mit* $P \sqsubseteq_{\text{as}} Q$*, ist* P *eine* as-Implementierung *von* Q *und es wird* as-impl(Q) *für die Menge aller as-Implementierungen von* Q *verwendet.*

Da für zwei MEIOs P und Q und alle möglichen Zustands-Tupel (p, q) in einer alternierenden Simulations-Relation $\mathcal{R}$ gelten muss, dass aus $q \notin E_Q$ folgt, dass auch p nicht in E_P enthalten ist, gilt auch die Implikation $p \in E_P \Rightarrow q \in E_Q$.

Definition 1.4 (*schwache alternierende Simulation*). *Eine* schwache alternierende Simulation *ist eine Relation* $\mathcal{R} \subseteq P \times Q$ *auf zwei MEIOs* P *und* Q*, wenn für alle* $(p, q) \in \mathcal{R}$ *mit* $q \notin E_Q$ *und* $i \in I$ *und* $\omega \in O \cup \{\tau\}$ *gilt:*

1. $p \notin E_P$*,*
2. $q \xrightarrow{i}_Q q'$ *impliziert* $p \xrightarrow{i}_P \overset{\varepsilon}{\Longrightarrow}_P p'$ *für ein* p' *mit* $p' \mathcal{R} q'$*,*

3. $q \xrightarrow{\omega}_Q q'$ *impliziert* $p \overset{\hat{\omega}}{\Longrightarrow}_P p'$ *für ein* p' *mit* $p' \mathcal{R} q'$,

4. $p \overset{i}{\dashrightarrow}_P p'$ *impliziert* $q \overset{i}{\dashrightarrow}_Q \overset{\varepsilon}{\Longrightarrow}_Q q'$ *für ein* q' *mit* $p' \mathcal{R} q'$,

5. $p \overset{\omega}{\dashrightarrow}_P p'$ *impliziert* $q \overset{\hat{\omega}}{\Longrightarrow}_Q q'$ *für ein* q' *mit* $p' \mathcal{R} q'$.

Analog zur starken alternierenden Simulation, wird hier $\sqsubseteq_{\text{w-as}}$ *als Relationssymbol verwendet und man kann auch entsprechend* schwache as-Verfeinerungen *betrachten.*
Ebenso kann $\sqsubseteq_{\text{w-as}}$ *für ein MEIO* Q *und eine Implementierung* P *definiert werden mit* $P \sqsubseteq_{\text{w-as}} Q$*, ist* P *eine* w-as-Implementierung *von* Q *und es wird* w-as-impl(Q) *für die Menge aller w-as-Implementierungen von* Q *verwendet.*

Die schwache Simulation erlaubt interne Aktionen beim MEIO, das die entsprechende Aktion matchen muss. Jedoch ist es zwingen notwendig, dass ein Input sofort aufgeführt wird und erst dann interne Aktinen möglich sind. Da ein Input die Reaktion auf eine Aktion ist, die die Umwelt auslöst und die nicht auf das Transitionssystem warten kann. Outputs hingegen können auch verzögert werden, da die Umgebung dies dann als Inputs aufnimmt und für diese somit nicht lokal kontrolliert ist.
Auch für alle Tupel (p, q) in einer schwach alternierenden Simulations-Relation $\mathcal{R}$ gilt $p \in E_P \Rightarrow q \in E_Q$.

Die Parallelkomposition von Wörtern und Mengen kann z.B. aus [BV15a] übernommen werden.

Definition 1.5 (*Parallelkomposition auf Traces*).

- *Für zwei Wörter* $w_1 \in \Sigma_1$ *und* $w_2 \in \Sigma_2$ *ist deren Parallelkomposition definiert als:* $w_1 \| w_2 := \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid w|_{\Sigma_1} = w_1 \wedge w|_{\Sigma_2} = w_2\}$.
- *Für zwei Mengen von Wörtern bzw. Sprachen* $W_1 \subseteq \Sigma_1^*$ *und* $W_2 \subseteq \Sigma_2^*$ *ist deren Parallelkomposition definiert als:* $W_1 \| W_2 := \bigcup \{w_1 \| w_2 \mid w_1 \in W_1 \wedge w_2 \in W_2\}$.

Ebenso können die Definitionen der Funktionen prune und cont zum Abschneiden und Verlängern von Traces z.B. aus [BV15a] übernommen werden. Hierbei ist zu beachten, dass in dieser Arbeit ε das leere Wort und $\mathfrak{P}(M)$ die Potenzmenge der Menge M bezeichnet.

Definition 1.6 (*Pruning- und Fortsetzungs-Funktion*).

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

Definition 1.7 (*Sprache*). *Die* Sprache *eines MEIOs* P *ist definiert als:* $L(P) := \left\{w \in \Sigma^* \mid p_0 \overset{w}{\Longrightarrow}_P\right\}$.

Somit entspricht die Sprache eines MEIOs, das eine Implementierung ist der Definition aus [Sch16] für EIOs. Jedoch muss die Sprache einer as-Verfeinerung eine MEIOs nicht mehr Teilmenge der Sprache des MEIOs sein, da Definition 1.3 beliebiges Verhalten nach einem Fehler-Zustand in dem zu verfeinernden MEIO zulässt. Falls jedoch der MEIO bereits fehler-frei ist, ist seine Sprache die Vereinigung der Sprachen all seine möglichen as-Implementierungen.
Von der Sprache einer as-Verfeinerung eines MEIOs kann man keine Rückschlüsse mehr auf die ursprüngliche Sprache ziehen, da man nicht weiß, welche Fehler-Zustände in der Verfeinerung übernommen wurden und welche als normale Zustände mit beliebigen Verhalten umgesetzt wurden.
Man hätte alternativ die Sprache eines MEIOs anders definieren können um weiterhin einen eindeutigen Zusammenhang zwischen dieser und den Sprachen der as-Implementierungen zu haben, jedoch wäre dann die Äquivalenz zur EIO Sprach-Definition in [Sch16] verloren gegangen. Eine Implementierung mit Fehler-Zuständen hätte dann auch eine Sprache mit Wörtern, die sie nicht mal ausführen können muss.

2 allgemeine Folgerungen

Proposition 2.1 (*Sprache und Implementierungen*). *Für die Sprache eines MEIOs P gilt* $L(P) \subseteq \left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}_{P'}\right\} = \bigcup_{P' \in \text{as-impl}(P)} L(P')$.

Beweis.
Sei P' die as-Implementierung von P, die alle may- und must-Transitionen von P implementiert. Die entsprechende starke as-Verfeinerungs-Relation $\mathcal{R}$, die zwischen P' und P gilt, ist die Identitäts-Relation zwischen den Zuständen der Transitionssysteme. Die Definition von P' lautet dann:

- $P' = P$,
- $p'_0 = p_0$,
- $I_{P'} = I_P$ und $O_{P'} = O_P$,
- $\longrightarrow_{P'} = \dashrightarrow_{P'} = \dashrightarrow_P$,
- $E'_P = \emptyset$.

Für alle $p \in P$, die auch von nicht Fehler-Zuständen aus erreichbar sind, muss $p\mathcal{R}p'$ wegen 1.3 3. für entsprechenden p' in P', die durch die analogen Transitionen erreichbar sind. Diese Tupel sind bereits in der Identitäts-Relation, die als as-Verfeinerungs-Relation vorausgesetzt wurde, enthalten. Da P' alle Transitionen von P implementiert, gilt 1.3 2. bereits durch die Tupel, die durch die Identitäts-Relation in $\mathcal{R}$ enthalten sein müssen. Der erste Punkt von 1.3 gilt, da E'_P leer ist. Für alle $w \in L(P) = \left\{w \in \Sigma^* \mid p_0 \overset{w}{\Longrightarrow}_P\right\}$ folgt nun $w \in L(P') = \left\{w \in \Sigma^* \mid p'_0 \overset{w}{\Longrightarrow}_{P'}\right\}$, da alle Transitionen von P in P' implementiert werden. □

Proposition 2.2 (*Sprache der Parallelkomposition*). *Für zwei komponierbare MEIOs P_1 und P_2 gilt: $L_{12} := L(P_{12}) = L_1 \| L_2$.*

Beweis. Jedes Wort, dass in L_{12} enthalten ist, hat einen einen entsprechenden Ablauf, der in P_{12} ausführbar ist. Dieser Ablauf kann auf Abläufe von P_1 und P_2 projiziert werden und die Projektionen sind dann in L_1 und L_2 enthalten.
In einer Parallelkomposition werden die Wörter der beiden MEIOs gemeinsam ausgeführt, falls es sich um synchronisierte Aktionen handelt, und verschränkt sequenziell, wenn es sich um unsynchronisierte Aktionen handelt. Somit sind alle Wörter aus $L_1 \| L_2$ auch Wörter der Parallelkomposition $L(P_{12})$. □

Lemma 2.3 (*w-as-Verfeinerung und Parallelkomposition*). *Für zwei komponierbar MEIOs P_1 und P_2 gilt, falls P_1' und P_2' schwache as-Verfeinerungen von P_1 bzw. P_2 sind, dann ist auch $P_1'\|P_2'$ eine schwache as-Verfeinerung von $P_1\|P_2$.*

Beweis. Es gelte $j \in \{1,2\}$. Da P_j' eine schwache as-Verfeinerung von P_j ist, gibt es nach Definition 1.4 eine schwache as-Verfeinerungs-Relation $\mathcal{R}_j$, die beschreibt, wie P_j' P_j verfeinert. Die Parallelkompositionen werden auf Basis von Definition 1.2 gebildet. Die Zustände sind also Tupel der Zustände der Komponenten. In dem man aus den Zuständen, die die $\mathcal{R}_j$ in Relation setzt auch solche Tupel zusammensetzt, kann man eine neue schwache as-Verfeinerungs-Relation für die Verfeinerung von $P_1\|P_2$ durch $P_1'\|P_2'$ erstellen. Die neue schwache as-Verfeinerungs-Relation soll $\mathcal{R}_{12}$ heißen und wie folgt definiert sein: $\forall p_1', p_2', p_1, p_2 : ((p_1', p_2'), (p_1, p_2)) \in \mathcal{R}_{12} \Leftrightarrow (p_1', p_1) \in \mathcal{R}_1 \wedge (p_2', p_2) \in \mathcal{R}_2$. Es bleibt nun zu zeigen, dass $\mathcal{R}_{12}$ eine zulässige schwache as-Verfeinerungs-Relation nach Definition 1.4 ist. Für alle folgenden Fälle wird $((p_1', p_2'), (p_1, p_2)) \in \mathcal{R}_{12}$ mit $(p_1, p_2) \notin E_{12}$ vorausgesetzt.

1. Für den ersten Punkt ist zu zeigen, dass (p_1', p_2') kein Element von $E_{P_1'\|P_2'}$ ist.
   Dies folgt direkt aus der Voraussetzung, dass $(p_1, p_2) \notin E_{P_1\|P_2}$ für das Tupel $((p_1', p_2'), (p_1, p_2))$ aus $\mathcal{R}_{12}$ gilt. In dem man auf das $\mathcal{R}_{12}$ die Definition von oben anwendet, erhält man $(p_j', p_j) \in \mathcal{R}_j$ für beide j Werte. Die p_j dürfen beide keine Fehler-Zustände sein, da sonst auch (p_1, p_2) ein solcher wäre. Somit folgt mit Definition 1.4 1. $p_j' \notin E_j$ für beide j Werte. Die beiden gestrichenen Zustände in Parallelkomposition können also keinen geerbten Fehler produzieren. Jedoch könnte (p_1', p_2') aufgrund eines nicht erzwungenen Inputs ein neuer Fehler-Zustand sein. Dafür müsste oBdA $p_1' \overset{a}{\not\longrightarrow}_{P_1'}$ und $p_2' \overset{a}{\dashrightarrow}_{P_2'}$ für ein a aus $I_1 \cap O_2$ gelten. $\mathcal{R}_2$ erzwingt mit 1.4 5. die schwache Ausführbarkeit des Outputs a in P_2, d.h. $p_2 \overset{a}{\Longrightarrow}_2$. Mit 1.4 2. von $\mathcal{R}_1$ folgt $p_1 \overset{a}{\not\longrightarrow}_1$. Somit müsste auch $(p_1, p_2) \in E_{12}$ gelten, was ein Widerspruch zur Voraussetzung wäre. (p_1', p_2') kann also weder ein geerbter noch ein neuer Fehler-Zustand in $P_1'\|P_2'$ sein und deshalb gilt $(p_1', p_2') \notin E_{P_1'\|P_2'}$.

2. Aus der Definition der schwachen alternierenden Simulation in 1.4 folgt, dass für diesen Punkt zu zeigen ist: $(p_1, p_2) \xrightarrow{i}_{12} (q_1, q_2)$ impliziert $(p_1', p_2') \xrightarrow{i}_{P_1'\|P_2'} \overset{\varepsilon}{\Longrightarrow}_{P_1'\|P_2'} (q_1', q_2')$ für ein (q_1', q_2') mit $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$.
   Die i-must-Transition in $P_1\|P_2$ kann entweder aus der Synchronisation von zwei must-Inputs entstanden sein oder als unsynchronisierte Aktion aus einem P_1 übernommen worden sein.

   - Fall 1 ($i \notin \text{Synch}(P_1\|P_2)$): OBdA ist i in I_1 enthalten. Es muss also in P_1 die i-Transition als must-Transition von p_1 ausgehen, es gilt $p_1 \xrightarrow{i}_1 q_1$. Mit der Relation $\mathcal{R}_1$ und 1.4 2. folgt, dass in P_1' i als schwache Transition in der Form $p_1' \xrightarrow{i}_{P_1'} \overset{\varepsilon}{\Longrightarrow}_{P_1'} q_1'$ ausführbar sein musst und $q_1' \mathcal{R}_1 q_1$ gelten muss. $p_2 = q_2$ muss gelten, da i nicht in Σ_2 enthalten ist. Aus der Voraussetzung folgt $(p_2', p_2) = (q_2', q_2) \in \mathcal{R}_2$, da i wenn es kein Element der Aktionen von P_2 ist auch keine Aktion der schwachen as-Verfeinerung P_2' sein kann. Mit der

Definition von $\mathcal{R}_{12}$ kann dann daraus $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ gefolgert werden. In der Parallelkomposition von P_1' und P_2' entsteht die Transitionsfolge $(p_1', p_2') \xrightarrow{i}_{P_1' \| P_2'} \xRightarrow{\varepsilon}_{P_1' \| P_2'} (q_1', q_2')$.

- Fall 2 ($i \in \text{Synch}(P_1 \| P_2)$): Damit i auch in $P_1 \| P_2$ ein Input ist, muss $i \in I_1 \cap I_2$ gelten. Um die Transition $(p_1, p_2) \xrightarrow{i}_{12} (q_1, q_2)$ in der Komposition möglich zu machen, muss in beiden P_j's $p_j \xrightarrow{i}_j q_j$ gelten. Durch $\mathcal{R}_j$ und die Definition 1.4 2. folgt für beide j Werte $p_j' \xrightarrow{i}_{P_j'} \xRightarrow{\varepsilon}_{P_j'} q_j'$ mit $(q_j', q_j) \in \mathcal{R}_j$. Daraus ergibt sich $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ mit der Definition von $\mathcal{R}_{12}$. Durch die Synchronisation der i-Inputs in der Komposition von P_1' und P_2' gilt $(p_1', p_2') \xrightarrow{i}_{P_1' \| P_2'} \xRightarrow{\varepsilon}_{P_1' \| P_2'} (q_1', q_2')$.

3. Analog zu 2. kann für diesen Punkt $(p_1, p_2) \xrightarrow{\omega}_{12} (q_1, q_2)$ impliziert $(p_1', p_2') \xRightarrow{\hat{\omega}}_{P_1' \| P_2'} (q_1', q_2')$ für ein (q_1', q_2') mit $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ gezeigt werden.
   Die ω Transition in $P_1 \| P_2$ ist entweder aus einem synchronisierten oder aus einem unsynchronisierten ω entstanden.

   - Fall 1 ($\omega \notin \text{Synch}(P_1 \| P_2)$): OBdA ist ω in $O_1 \cup \{\tau\}$ enthalten. Um in der Komposition $P_1 \| P_2$ die must-Transition zu erhalten muss bereits für die Transition in P_1 $p_1 \xrightarrow{\omega}_1 q_1$ gelten. Mit 1.4 3. kann für $\mathcal{R}_1$ gefolgert werden, dass $p_1' \xRightarrow{\hat{\omega}}_{P_1'} q_1'$ mit $(q_1', q_1) \in \mathcal{R}_1$ gilt. In der Komposition folgt dann $(p_1', p_2') \xRightarrow{\hat{\omega}}_{P_1' \| P_2'} (q_1', q_2')$, da $\omega \notin \Sigma_2$ ist und somit $(p_2', p_2) = (q_2', q_2) \in \mathcal{R}_2$ gilt. Es folgt insgesamt auch noch die Zugehörigkeit des Zustands-Tupels $((q_1', q_2'), (q_1, q_2))$ zur schwachen as-Verfeinerungs-Relation $\mathcal{R}_{12}$.

   - Fall 2 ($\omega \in \text{Synch}(P_1 \| P_2)$): Da in der Menge $\text{Synch}(P_1 \| P_2)$ nur Inputs und Outputs enthalten sein können, muss in diesem Fall $\omega \neq \tau$ gelten. Um einen Output ω in der Parallelkomposition von P_1 und P_2 zu erhalten, muss oBdA $\omega \in I_1 \cap O_2$ gelten. Es folgt also $p_1 \xrightarrow{\omega}_1 q_1$ mit $\omega \in I_1$ und $p_2 \xrightarrow{\omega}_2 q_2$ mit $\omega \in O_2$ für die einzeln Transitionssysteme. Mit $\mathcal{R}_1$ und 1.4 2. folgt $p_1' \xrightarrow{\omega}_{P_1'} \xRightarrow{\varepsilon}_{P_1'} q_1'$ und $q_1' \mathcal{R}_1 q_1$. Wenn man $\mathcal{R}_2$ mit 1.4 3. angewendet erhält man $p_2' \xRightarrow{\omega}_{P_2'} q_2'$ mit $q_2' \mathcal{R}_2 q_2$. Da ω in P_2' ein Output ist, gilt $\omega = \hat{\omega}$. In der Parallelkomposition von P_1' und P_2' werden zuerst die internen Aktionen von P_2' ausgeführt, bis dort der Output erreicht ist, dann wird ω synchronisiert und danach werden die internen Aktionen beider Komponenten ausgeführt, bis man bei den Zuständen q_1' und q_2' angekommen ist. Es folgt also die Transitionsfolge $(p_1', p_2') \xRightarrow{\hat{\omega}}_{P_1' \| P_2'} (q_1', q_2')$ und das Tupel $((q_1', q_2'), (q_1, q_2))$ in der Relation $\mathcal{R}_{12}$.

4. $(p_1', p_2') \xrightarrow{i}_{P_1' \| P_2'} (q_1', q_2')$ impliziert $(p_1, p_2) \xrightarrow{i}_{12} \xRightarrow{\varepsilon}_{12} (q_1, q_2)$ für ein (q_1, q_2) mit $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ ist die Voraussetzung des 4. Punktes, um zu beweisen, dass $\mathcal{R}_{12}$ eine schwache as-Verfeinerungs-Relation ist.

Die Transition i kann wiederum durch Synchronisation von zwei Transitionen entstanden sein oder durch eine Transition aus einer der beiden Komponenten mit der Voraussetzung $i \notin \text{Synch}(P_1' \| P_2')$.

- Fall 1 ($i \notin \text{Synch}(P_1' \| P_2')$): OBdA ist i in I_1 enthalten. Es muss also in P_1' eine ausgehende i-Transition von Zustand p_1' geben, so dass $p_1' \overset{i}{\dashrightarrow}_1 q_1'$ gilt. Mit der Relation $\mathcal{R}_1$ und 1.4 4. folgt, dass in P_1 i als schwache Transition in der Form $p_1 \overset{i}{\dashrightarrow}_1 \overset{\varepsilon}{=\!\Rightarrow}_1 q_1$ ausführbar sein musst und $q_1' \mathcal{R}_1 q_1$ gelten muss. $p_2' = q_2'$ muss gelten, da i nicht in Σ_2 enthalten ist. Aus der Voraussetzung folgt $(p_2', p_2) = (q_2', q_2) \in \mathcal{R}_2$, da i wenn es kein Element der Aktionen von P_2' ist auch keine Aktion der Spezifikation P_2 sein kann. Mit der Definition von $\mathcal{R}_{12}$ kann dann daraus $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ gefolgert werden. In der Parallelkomposition von P_1 und P_2 entsteht die Transitionsfolge $(p_1, p_2) \overset{i}{\dashrightarrow}_{12} \overset{\varepsilon}{=\!\Rightarrow}_{12} (q_1, q_2)$.
- Fall 2 ($i \in \text{Synch}(P_1' \| P_2')$): Damit i auch in $P_1' \| P_2'$ ein Input ist, muss $i \in I_1 \cap I_2$ gelten. Um die Transition $(p_1', p_2') \overset{i}{\dashrightarrow}_{P_1' \| P_2'} (q_1', q_2')$ in der Komposition möglich zu machen, muss in beiden Transitionssystemen P_j' $p_j \overset{i}{\dashrightarrow}_{P_j'} q_j'$ gelten. Durch $\mathcal{R}_j$ und die Definition 1.4 4., die für diese Relationen gilt, folgt für beide j Werte $p_j \overset{i}{\dashrightarrow}_j \overset{\varepsilon}{=\!\Rightarrow}_j q_j$ mit $(q_j', q_j) \in \mathcal{R}_j$. Es folgt $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ mit der Definition von $\mathcal{R}_{12}$. Durch die Synchronisation des i's in der Komposition von P_1 und P_2 gilt $(p_1, p_2) \overset{i}{\dashrightarrow}_{12} \overset{\varepsilon}{=\!\Rightarrow}_{12} (q_1, q_2)$.

5. Analog zu 3. und 4. kann für diesen Punkt $(p_1', p_2') \overset{\omega}{\dashrightarrow}_{P_1' \| P_2'} (q_1', q_2')$ impliziert $(p_1, p_2) \overset{\hat{\omega}}{=\!\Rightarrow}_{12} (q_1, q_2)$ für ein (q_1, q_2) mit $((q_1', q_2'), (q_1, q_2)) \in \mathcal{R}_{12}$ gezeigt werden.
   Die ω Transition in $P_1' \| P_2'$ ist entweder aus einem synchronisierten oder aus einem unsynchronisierten ω entstanden.
   - Fall 1 ($\omega \notin \text{Synch}(P_1' \| P_2')$): OBdA ist ω in $O_1 \cup \{\tau\}$ enthalten. Um in $P_1' \| P_2'$ die may-Transition zu erhalten muss bereits in P_1' die Transition $p_1' \overset{\omega}{\dashrightarrow}_{P_1'} q_1'$ möglich gewesen sein. Mit 1.4 5. kann für $\mathcal{R}_1$ gefolgert werden, dass $p_1 \overset{\hat{\omega}}{=\!\Rightarrow}_1 q_1$ mit $(q_1', q_1) \in \mathcal{R}_1$ gilt. In der Komposition folgt dann $(p_1, p_2) \overset{\hat{\omega}}{=\!\Rightarrow}_{12} (q_1, q_2)$, da $\omega \notin \Sigma_2$ ist und somit $(p_2', p_2) = (q_2', q_2) \in \mathcal{R}_2$ gilt. Es folgt insgesamt auch noch die Zugehörigkeit des Zustands-Tupels $((q_1', q_2'), (q_1, q_2))$ zur schwachen as-Verfeinerungs-Relation $\mathcal{R}_{12}$.
   - Fall 2 ($\omega \in \text{Synch}(P_1' \| P_2')$): Es muss $\omega \neq \tau$ gelten und somit muss oBdA $\omega \in I_1 \cap O_2$ gelten. Es folgt also $p_1' \overset{\omega}{\dashrightarrow}_{P_1'} q_1'$ mit $\omega \in I_1$ und $p_2' \overset{\omega}{\dashrightarrow}_{P_2'} q_2'$ mit $\omega \in O_2$ für die einzeln Transitionssysteme. Mit $\mathcal{R}_1$ und 1.4 4. folgt $p_1 \overset{\omega}{\dashrightarrow}_1 \overset{\varepsilon}{=\!\Rightarrow}_1 q_1$ und $q_1' \mathcal{R}_1 q_1$. Wenn man $\mathcal{R}_2$ mit 1.4 5. angewendet erhält man $p_2 \overset{\omega}{=\!\Rightarrow}_2 q_2$ mit $q_2' \mathcal{R}_2 q_2$. Da ω in P_2 ein Output ist, gilt $\omega = \hat{\omega}$. In der Parallelkomposition von P_1 und P_2 werden zuerst die internen Aktionen von P_2 ausgeführt, bis dort der Output erreicht ist, dann wird ω synchronisiert und danach werden die internen Aktionen beider Komponenten ausgeführt, bis

man bei den Zuständen q_1 und q_2 angekommen ist. Es folgt also die Transitionsfolge $(p_1, p_2) \stackrel{\hat{\omega}}{\Rightarrow}_{12} (q_1, q_2)$ und das Tupel $((q_1', q_2'), (q_1, q_2))$ in der Relation $\mathcal{R}_{12}$.

□

Korollar 2.4 (*w-as-Implementierungen und Parallelkomposition*). *Für zwei komponierbare MEIOs P_1 und P_2 gilt: $P_1' \in$ w-as-impl$(P_1) \wedge P_2' \in$ w-as-impl$(P_2) \Rightarrow (P_1' \| P_2') \in$ w-as-impl$(P_1 \| P_2)$.*

Beweis. P_1' und P_2' sind Aufgrund der Definition 1.4 auch schwache as-Verfeinerungen von P_1 bzw. P_2. Somit ist die Parallelkomposition $P_1' \| P_2'$ auch eine schwache as-Verfeinerung von $P_1 \| P_2$, wegen Lemma 2.3. Für Implementierungen gilt $\longrightarrow = \dashrightarrow$. Durch die Definition der Parallelkomposition in 1.2 können aus aus zwei komponierbaren Implementierungen in der Komposition keine may-Transitionen ohne zugehörige must-Transitionen entstehen. Es gilt also auch $\longrightarrow_{P_1' \| P_2'} = \dashrightarrow_{P_1' \| P_2'}$ und somit ist $P_1' \| P_2'$ eine Implementierung und eine as-Verfeinerung von $P_1 \| P_2$. Dies entspricht der Definition der schwachen as-Implementierung, sodass $(P_1' \| P_2') \in$ w-as-impl$(P_1 \| P_2)$ gilt. □

Korollar 2.5 (*as-Verfeinerungen und Parallelkomposition*). *Für zwei komponierbar MEIOs P_1 und P_2 gilt, falls P_1' und P_2' as-Verfeinerungen von P_1 bzw. P_2 sind, dann ist auch $P_1' \| P_2'$ eine as-Verfeinerung von $P_1 \| P_2$.*

Beweis. Falls die Relationen $\mathcal{R}_1$ und $\mathcal{R}_2$ aus dem Beweis von Lemma 2.3 keine schwachen as-Verfeinerungs-Relationen sondern starke as-Verfeinerungs-Relation sind, ist auch $\mathcal{R}_{12}$ eine starke as-Verfeinerungs-Relation zwischen $P_1' \| P_2'$ und $P_1 \| P_2$. Es ist also nur zu zeigen, wie aus den einzelnen Beweispunkten des Beweises von 2.3 folgt, dass $\mathcal{R}_{12}$ eine starke as-Verfeinerungs-Relation ist. Es gilt hier ebenso für alle Punkte die Voraussetzung $((p_1', p_2'), (p_1, p_2)) \in \mathcal{R}_{12}$ mit $(p_1, p_2) \notin E_{12}$.

1. Dieser Punkt kann genauso wie 1. aus dem Beweis von Lemma 2.3 bewiesen werden. Nur für p_2 ist der Output a nicht nur schwach ausführbar, sondern direkt.

2. α kann sowohl Input, Output wie auch interen Aktion sein. Um diesen Punkt zu beweisen muss man 2. und 3. aus dem Beweis von Lemma 2.3 kombinieren. Da die $\mathcal{R}_j$ für $j \in \{1, 2\}$ jedoch die Transition in den P_j' ohne zusätzliche τ-Transitionen fordern, entstehen in den einzelnen Komponenten keine schwachen Transitionen für die αs und somit ist α auch in der Parallelkomposition $P_1' \| P_2'$ eine direkte Transition ohne zusätzliche τs. Es folgt also das zu zeigende für diesen die strake as-Verfeinerungs-Relation für diesen Punkt.

3. Hierfür werden die Punkte 3. und 4. aus dem Beweis des Lemmas 2.3 kombiniert. Analog wie bei 2. diese Beweises fallen die zusätzlichen τ-Transitionen durch die stärkere Forderung an $\mathcal{R}_1$ und $\mathcal{R}_2$ weg. Dieser Punkt gilt also ebenfalls.

□

Korollar 2.6 (*as-Implementierungen und Parallelkomposition*). *Für zwei komponierbare MEIOs P_1 und P_2 gilt:* $P_1' \in \text{as-impl}(P_1) \wedge P_2' \in \text{as-impl}(P_2) \Rightarrow (P_1' \| P_2') \in \text{as-impl}(P_1 \| P_2)$.

Beweis. Analog zum Beweis des Korollars 2.4 kann hier auch begründet werden, dass $P_1' \| P_2'$ eine Implementierung ist. Zusätzlich ist $P_1' \| P_2'$ eine as-Verfeinerung , wegen der Voraussetzungen diese Korollars in Kombination mit der Aussage des Korollars 2.5. Es gilt also $(P_1' \| P_2') \in \text{as-impl}(P_1 \| P_2)$. □

Die entgegengesetzte Richtung von Korollar 2.5 gilt im allgemeinen nicht, d.h. es muss zu einer as-Verfeinerung P' einer Parallelkomposition $P_1 \| P_2$ keine as-Verfeinerungen P_1' bzw. P_2' der einzelnen Komponenten P_1 bzw. P_2 geben, deren Parallelkomposition $P_1' \| P_2'$ der as-Verfeinerung der Parallelkomposition P' entsprechen. Die Problematik wird in Abbildung 2.1 an einem Beispiel dargestellt. In der Parallelkomposition wird die may-Transition von P_2 zu zwei may-Transitionen, für die in einer as-Verfeinerung unabhängig entschieden werden kann, ob sie übernommen, implementiert oder weggelassen werden. Somit kommt es in P' zu dem Problem, dass keine as-Verfeinerung von P_2 (entweder keine Transition oder die o' Transition wird als may- oder must-Transition ausgeführt) in Parallelkomposition mit der Implementierung P_1 P' ergeben würde.
Da jede as-Verfeinerungs-Relation auch eine schwache as-Verfeinerungs-Relation ist, folgt draus auch, dass die entgegengesetzt Richtung von Lemma 2.3 ebenfalls nicht gelten kann. Auch im Spezialfall von as-Implementierungen bzw. w-as-Implementierungen kann das Gegenbeispiel angewendet werden, da P' auch eine Implementierung von $P_1 \| P_2$ ist und es auch keine passende as-Implementierung bzw. w-as-Implementierung von P_2 geben kann, wenn es schon keine passende Verfeinerung gibt.

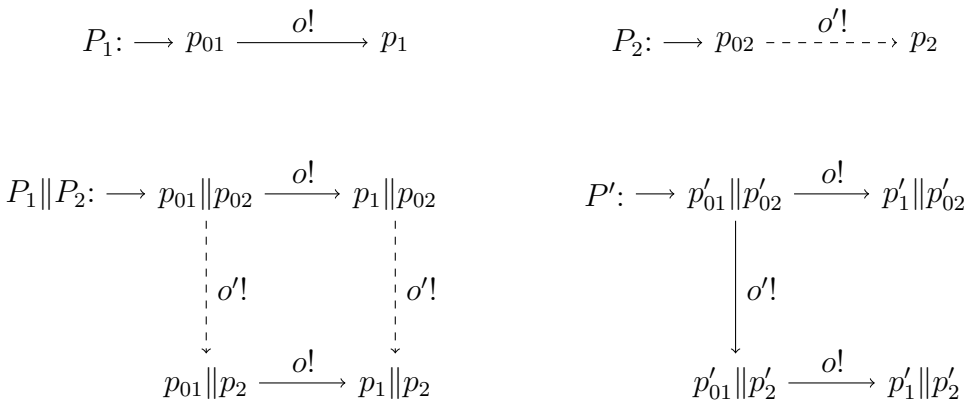

Abbildung 2.1: Gegenbeispiel für Umkehrung von Lemma 2.5

Ein neuer Fehler in einer Parallelkomposition zweier MEIOs muss in einer Implementierung (as oder w-as) dieser Parallelkomposition nicht auftauchen, auch nicht in der Parallelkomposition von Implementierungen der einzelnen Komponenten. Dies liegt daran,

dass für den Input nur vorausgesetzt wird, dass keine must-Transition für die Synchronisation der Aktion vorhanden ist. Es kann trotzdem eine may-Transition für den Input geben, die auch implementiert werden kann. Falls es aber in der Parallelkomposition zweier MEIO zu einem neuen Fehler kommt, dann gibt es auch immer mindestens eine mögliche Implementierung, die diesen Fehler enthält und es gibt auch immer mindestens ein Implementierungs-Paar der Komponenten, in deren Parallelkomposition sich dieser Fehler ebenfalls zeigt.

3 Verfeinerungen für Kommunikationsfehler-Freiheit

3.1 Erweiterungs-Ansatz

Dieses Kapitel versucht die Präkongruenz für Error bei EIOs aus z.B. [Sch16] auf die hier betrachten MEIOs zu erweitern.

Definition 3.1 (*fehler-freie Kommunikation*). *Ein Fehler-Zustand ist* lokal erreichbar *in einem MEIO P, wenn ein $w \in O^*$ existiert mit $p_0 \overset{w}{\Rightarrow}_P p \in E$.*
Zwei MEIOs P_1 und P_2 kommunizieren fehler-frei*, wenn keine as-Implementierungen ihrer Parallelkomposition P_{12} einen Fehler-Zustände lokal erreichen kann.*

Definition 3.2 (*Kommunikationsfehler-Verfeinerungs-Basirelation*). *Für zwei MEIOs P_1 und P_2 mit der gleichen Signatur wird $P_1 \sqsubseteq_E^{\mathrm{B}} P_2$ geschrieben, wenn nur dann ein Fehler-Zustand in einer as-Implementierung von P_1 lokal erreichbar ist, wenn es auch eine as-Implementierung von P_2 gibt, in der ein Fehler-Zustand auch lokal erreichbar ist. Die Basisrelation stellt eine* Verfeinerung-Relation *bezüglich* Kommunikationsfehler-Freiheit *dar.*
$\sqsubseteq_E^{\mathrm{C}}$ bezeichnet die vollständig abstrakte Präkongruenz *von $\sqsubseteq_E^{\mathrm{B}}$ bezüglich $\cdot\|\cdot$, d.h. die gröbste Präkongruenz bezüglich $\cdot\|\cdot$, die in $\sqsubseteq_E^{\mathrm{B}}$ enthalten ist.*

Für as-Implementierungen P_1 und P_2 entspricht $\sqsubseteq_E^{\mathrm{B}}$ der Relation $\sqsubseteq_E^{\mathrm{B}}$ aus [Sch16].

Wie z.B. in [Sch16] werden die Fehler hier Trace-basiert betrachtet.

Definition 3.3 (*Kommunikationsfehler-Traces*). *Für ein MEIO P wird definiert:*

- strikte Fehler-Traces*:* $StET(P) := \left\{ w \in \Sigma^* \mid p_0 \overset{w}{\Rightarrow}_P p \in E \right\}$,
- gekürzte Fehler-Traces*:* $PrET(P) := \{\mathrm{prune}(w) \mid w \in StET(P)\}$,
- Input-kritische-Traces*:* $MIT(P) := \left\{ wa \in \Sigma^* \mid p_0 \overset{w}{\Rightarrow}_P p \wedge a \in I \wedge p \overset{a}{\not\rightarrow}_P \right\}$.

Da die Basisrelation über as-Implementierungen spricht, ist es wichtig bereits in den Trace-Mengen eine Beziehung zwischen der allgemeinen Definition für MEIOs und deren as-Implementierungen herzustellen. Die nächste Proposition stellt eine Teilmengen-Beziehung zwischen den Traces eines MEIOs und den Traces seiner as-Implementierungen dar.

Proposition 3.4 (*Kommunikationsfehler-Traces und Implementierungen*). *Sei P ein MEIO.*

1. *Für die strikten Fehler-Traces von P gilt:* $StET(P) \subseteq \left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}_{P'} p' \in E\right\} = \bigcup_{P' \in \text{as-impl}(P)} StET(P')$. TODO: erzwungen Zeilenumbruch kontrollieren

2. *Für die gekürzten Fehler-Traces von P gilt:* $PrET(P) \subseteq \{\text{prune}(w) \mid \exists P' \in \text{as-impl}(P) : w \in StET(P')\} = \bigcup_{P' \in \text{as-impl}(P)} PrET(P')$, TODO: erzwungen Zeilenumbruch kontrollieren

3. *Für Input-kritischen-Traces von P gilt:* $MIT(P) \subseteq \left\{wa \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}_{P'} p' \wedge a \in I \wedge p' \overset{a}{\not\longrightarrow}_{P'}\right\} = \bigcup_{P' \in \text{as-impl}(P)} MIT(P')$. TODO: erzwungen Zeilenumbruch kontrollieren

Beweis.

1. Um die Inklusion zu zeigen wird eine Implementierung P' angegeben, die die strikten Fehler-Traces von P implementiert und zusätzlich auch noch eine passende as-Verfeinerungs-Relation $\mathcal{R}$ zwischen den beiden Transitionssystemen. P' implementiert wie im Beweis zu Proposition 2.1 alle Transitionen von P. Das P' wird hier jedoch im Gegensatz zu Beweis von 2.1 auch noch alle Fehler-Zustände aus P implementieren. Die entsprechende as-Verfeinerungs-Relation $\mathcal{R}$ ist jedoch hier ebenfalls wieder die Identitäts-Relation zwischen den Zuständen der Transitionssysteme. Die Definition von P' lautet also:

   - $P' = P$,
   - $p'_0 = p_0$,
   - $I_{P'} = I_P$ und $O_{P'} = O_P$,
   - $\longrightarrow_{P'} = \dashrightarrow_{P'} = \dashrightarrow_P$,
   - $E'_P = E_P$.

   Die Tupel, die von 1.3 2. und 3. als Elemente der as-Verfeinerungs-Relation $\mathcal{R}$ gefordert werden, sind bereits durch die Identitäts-Relation garantiert. Wie im Beweis 2.1. Für 1.3 1. muss für jedes in der as-Verfeinerungs-Relation $\mathcal{R}$ enthaltene Zustands-Paar gelten, wenn der Zustand aus P kein Fehler-Zustand ist, dann ist auch der Zustand aus P' keiner, dies folgt aus der Gleichheit der Mengen E_P und E'_P. Jeder Trace aus $StET(P)$ ist via may-Transitionen in P ausführbar und führt dort zu einem Fehler-Zustand. Der analoge Trace ist auch in P' möglich, da alle may-Transitionen aus P in P' als must-Transitionen implementiert wurden. Der dabei erreichte Zustand steht mit dem Fehler-Zustand in P in der Identitäts-Relation $\mathcal{R}$, die Zustände entsprechen sich also. Es gilt mit $E'_P = E_P$, dass auch der in P' erreichte Zustand ein Fehler-Zustand ist. Für die as-Implementierung P' von P und der Identitäts-Relation $\mathcal{R}$ als starke as-Verfeinerungs-Relation zwischen den Transitionssystemen gilt also $StET(P) = StET(P')$.

2. Da der erste Punkt dieser Proposition bereits beweisen wurde, gilt bereits, dass alle strikten Fehler-Traces von P in der Vereinigung aller strikten Fehler-Traces der as-Implementierungen von P enthalten sind. Wenn auf alle Wörter in beiden Mengen die prune-Funktion angewendet wird, gilt die Inklusion der daraus entstanden Mengen weiterhin. Dies entspricht der Behauptung dies Punktes.

3. Auch für diese Inklusion wird eine starke as-Verfeinerungs-Relation $\mathcal{R}$ und eine Implementierung P' angegeben. Jedoch werden nicht wie bei 1. alle Transitionen von P in P' implementiert. Es wird auch für alle wa aus $MIT(P)$ eine eigene Implementierung P' geben und nicht eine für alle. Es werden alle must-Transitionen aus P in P' implementiert und zusätzlich die may-Transitionen, die zum Ausführen von w benötigt werden, so dass das a danach in P nicht gefordert wird. Es kann aufgrund möglicher Schleifen in P auch nicht mehr die Identitäts-Relation als as-Verfeinerungs-Relation gewählt werden. Der w Trace wird entsprechend seiner Länge abgewickelt, so dass sicher gestellt wird, dass der Zustand am Ende dieses Traces wirklich ruhig ist und nicht Teil einer Schlinge in w. Für das Abwickeln werden die Zustände entsprechend ihrer Position in w durchnummeriert. Für ein w, für das $wa \in MIT(P)$, gilt: $\exists w' \in \Sigma_\tau^*, \exists \alpha_1, \alpha_2, \ldots, \alpha_n, \exists p_1, p_2, \ldots, p_n : \hat{w'} = w \wedge w' = \alpha_1\alpha_2 \ldots \alpha_n \wedge p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \ldots p_{n-1} \overset{\alpha_n}{\dashrightarrow} p_n$. Die starke as-Verfeinerungs-Relation $\mathcal{R}$ enthält in diesem Fall Tupel $((p, j), p)$ für alle $0 \leq j \leq n$. Die entsprechende Definition für das P', das wa als Input-kritischen-Trace enthalten soll lautet:

   - $P' = P \times \{0, 1, \ldots n\}$,
   - $p'_0 = (p_0, 0)$,
   - $I'_P = I_P$ und $O'_P = O_P$,
   - $$\longrightarrow'_P = \dashrightarrow'_P = \left\{((p, j), \alpha, (p', j+1)) \mid p \overset{\alpha}{\dashrightarrow}_P p', 0 \leq j < n\right\} \cup \left\{((p, j), \alpha, (p', j)) \mid p \xrightarrow{\alpha}_P p', 0 \leq j \leq n\right\},$$
   - $E'_P = \emptyset$.

   Der Ablauf von w aus P wird in P' durch $(p_0, 0) \xrightarrow{\alpha_1} (p_1, 1) \xrightarrow{\alpha_1} \ldots (p_{n-1}, n-1) \xrightarrow{\alpha_n} (p_n, n)$ simuliert. w ist also in P' ausführbar. a ist für (p_n, n) nicht ausführbar, da in P für p_n $p_n \overset{a}{\not\longrightarrow}$ gilt und für die Zustände mit der Nummer n in P' nur die must-Transitionen implementiert werden. Die implementierten Transitionen in P' haben nach Definition alle zugrundeliegende may-Transitionen in P an den in Relation $\mathcal{R}$ stehenden Zuständen. Es werden durch die zweite Menge der Transitions-Definition von P' alle must-Transitionen aus P entsprechend in P' implementiert. Da die Menge E'_P leer ist, gelten alle Bedingungen, damit $\mathcal{R}$ eine as-Verfeinerungs-Relation zwischen P' und P ist. Mit der Begründung von oben folgt auch $wa \in MIT(P')$. Da für alle wa aus $MIT(P)$ eine entsprechende Implementierung mit as-Verfeinerungs-Relation $\mathcal{R}$ angegeben werden kann, gilt die Inklusion.

□

Definition 3.5 (*Kommunikationsfehler-Semantik*). *Sei P ein MEIO.*

- *Die Menge der* Fehler-Traces *von P ist* $ET(P) := \text{cont}(PrET(P)) \cup \text{cont}(MIT(P))$.
- *Die* Fehler-geflutete Sprache *von P ist* $EL(P) := L(P) \cup ET(P)$.

Für zwei MEIOs P_1, P_2 mit der gleichen Signatur wird $P_1 \sqsubseteq_E P_2$ geschrieben, wenn $ET_1 \subseteq ET_2$ und $EL_1 \subseteq EL_2$ gilt.

Hierbei ist zu beachten, dass die Mengen $StET$, $PrET$, MIT, ET und EL nur denen aus [Sch16] entsprechen, wenn P bereits eine as-Implementierung ist.
Aus den Propositionen für die Sprache und die Traces konnte für die Vereinigung der gleichen Mengen über die Implementierungen immer nur eine Inklusionsrichtung gefolgert werden, da die Definition 1.3 nach einen Fehler-Zustand in P beliebiges Verhalten ins dessen as-Implementierungen zulässt. Mit dem Einsatz der cont-Funktion zum beliebigen fortsetzten der Traces kann dies ausgeglichen werden. Somit gilt wie die nächsten Proposition behauptet für die Fehler-Traces und die Fehler-geflutete Sprache Gleichheit und nicht nur die Inklusion, die aus den Propositionen vorher bereits folgt.

Proposition 3.6 (*Kommunikationsfehler-Semantik und Implementierungen*). *Sie P ein MEIO.*

1. *Für die Menge der Fehler-Traces von P gilt* $ET(P) = \bigcup_{P' \in \text{as-impl}(P)} ET(P')$.
2. *Für die Fehler-geflutete Sprache von P gilt* $EL(P) = \bigcup_{P' \in \text{as-impl}(P)} EL(P')$.

Beweis.
1. „$\subseteq$“:

$$\begin{aligned} ET(P) &\overset{3.5}{=} \text{cont}(PrET(P)) \cup (MIT(P)) \\ &\overset{3.4}{\subseteq} \text{cont}\left(\bigcup_{P' \in \text{as-impl}(P)} PrET(P')\right) \cup \text{cont}\left(\bigcup_{P' \in \text{as-impl}(P)} MIT(P')\right) \\ &\overset{\substack{\text{cont} \\ \text{monoton}}}{=} \bigcup_{P' \in \text{as-impl}(P)} \text{cont}(PrET(P')) \cup \text{cont}(MIT(P')) \\ &\overset{3.5}{=} \bigcup_{P' \in \text{as-impl}(P)} ET(P'). \end{aligned}$$

1. „$\supseteq$“:
Da für P $ET(P) = \text{cont}(PrET(P)) \cup \text{cont}(MIT(P))$ gilt und für alle as-Implementierungen P' von P die analogen Gleichungen für $ET(P')$ gelten, genügt es ein präfix-minimales w aus $ET(P')$ für eine as-Implementierung P' von P zu betrachten. Da w präfix-minimal ist für P' ist w entweder vollständig oder bis auf den letzten Buchstaben

in P' ausführbar. Dies hängt davon ab, ob $w \in PrET(P')$ oder $w \in MIT(P')$ gilt. Für P muss jedoch nicht mal das Präfix von w ohne den letzten Buchstaben von w ausführbar sein. Für den weiteren Teil dieses Beweises soll gelten $w = va$ für $a \in \Sigma$ und $v \in \Sigma^*$.

- Fall 1 ($v \notin L(P)$): Das v kann in P nicht ausgeführt werden. Dies kann nur der Fall sein, wenn ein Präfix von v zu einem Fehler-Zustand in P führt, da sonst 1.3 3. verletzt wäre. Es ist also ein Präfix von w in $StET(P)$ enthalten und somit gilt wegen der cont-Funktion $w \in ET(P)$.

- Fall 2 ($v \in L(P)$ und $w \in PrET(P')$): In diesem Fall kann das Wort v in P ausgeführt werden. Falls a in P nicht ausgeführt werden kann, ist w aufgrund der gleichen Argumentation wie im 1. Fall in $ET(P)$ enthalten. Es wird also davon ausgegangen, dass $w \in L(P)$ gilt. Die Definition 1.3 fordert, dass alle Zustände p_j und p'_j aus P bzw. P' auf dem w-Trace in der starken as-Verfeinerungs-Relation $\mathcal{R}$ stehen. In P' existiert eine Verlängerung $v \in O^*$ von w, so dass $wv \in StET(P')$. Diese Verlängerung kann in P möglicherweise nicht ausführbar sein, dann ist jedoch analog zu Fall 1 ein Präfix von wv in $StET(P)$ enthalten. Da v nur aus Outputs besteht gilt $\text{prune}(wv) = \text{prune}(w)$. Es folgt also für ein Präfix von w die Zugehörigkeit zur Menge $PrET(P)$. Somit ist w in $ET(P)$ enthalten. Falls jedoch wv in P ausführbar, kann immer noch ein echtes Präfix von wv in P zu einem Fehler-Zustand führen, wodurch wieder $w \in ET(P)$ folgt. Falls jedoch alle Zustände (bis auf den letzten), die im Trace wv auftauchen keine Fehler-Zustände sind, gilt für p und p' mit $p_0 \overset{wv}{=\!\Rightarrow}_P p$ und $p'_0 \overset{wv}{\Longrightarrow}_{P'} p'$ $(p', p) \in \mathcal{R}$. Da $wv \in StET(P')$ gilt, muss $p' \in E_{P'}$ gelten. Daraus folgt mit 1.3 1., dass p in E_P enthalten sein muss. Es gilt also $wv \in StET(P)$. Mit $\text{prune}(w) = \text{prune}(wv)$ folgt daraus $w \in ET(P)$.

- Fall 3 ($v \in L(P)$ und $w \in MIT(P')$): Da w in $MIT(P')$ enthalten ist, $a \in I$. Falls ein Präfix von v oder v selbst in P zu einem Fehler Zustand führt, gilt wie in den beiden vorangegangen Fällen $w \in ET(P)$. Es wird also davon ausgegangen, das auf dem v Trace in P kein Fehler-Zustand erreicht wird auch der durch v erreichte Zustand p in P soll nicht in E_P enthalten sein. Es gilt also $p_0 \overset{v}{=\!\Rightarrow}_P p$ und $p'_0 \overset{v}{\Longrightarrow}'_P p'$ mit $(p', p) \in \mathcal{R}$. Wobei $\mathcal{R}$ die as-Verfeinerungs-Relation ist, die P' zu einer as-Implementierung von P macht. Falls a für p eine ausgehende must-Transition wäre, würde 1.3 2. auch für p' die Implementierung der a Transition fordern und somit würde nicht $wa \in MIT(P')$ gelten. Es gilt also $p \overset{a}{\not\longrightarrow}$ und $va \in MIT(P)$. Daraus ergibt sich direkt $w \in ET(P)$.

2. „$\subseteq$“:

$$
\begin{aligned}
EL(P) &\overset{3.5}{=} L(P) \cup ET(P) \\
&\overset{2.1}{\subseteq} \left(\bigcup_{P' \in \text{as-impl}(P)} L(P') \right) \cup ET(P) \\
&\overset{1.}{=} \left(\bigcup_{P' \in \text{as-impl}(P)} L(P') \right) \cup \left(\bigcup_{P' \in \text{as-impl}(P)} ET(P') \right)
\end{aligned}
$$

$$= \bigcup_{P' \in \text{as-impl}(P)} L(P') \cup ET(P')$$
$$\overset{3.5}{=} \bigcup_{P' \in \text{as-impl}(P)} EL(P').$$

2. „$\supseteq$“:
Da der erste Punkt dieser Proposition bereits beweisen ist, reicht es aus für diesen Punkt zu zeigen, dass $\bigcup_{P' \in \text{as-impl}(P)} EL(P') \backslash ET(P')$ eine Teilmenge von $EL(P)$ ist. Die Menge $EL(P') \backslash ET(P')$ entspricht $L(P') \backslash ET(P')$. Es musst also ein Wort w aus der Sprache einer as-Implementierung von P betrachtet werden, dass nicht in den Fehler-Traces dieser as-Implementierung enthalten ist. Das Wort w ist also in P' ausführbar. Falls das w in P jedoch nicht ausführbar ist, folgt wie zuvor $w \in ET(P) \subseteq EL(P)$, da ein Präfix von w in P zu einem Fehler-Zustand führen musst. Falls w ausführbar ist in P gilt $w \in L(P) \subseteq EL(P)$. □

Satz 3.7 (*Kommunikationsfehler-Semantik für Parallelkompositionen*). *Für zwei komponierbare MEIOs P_1, P_2 und ihre Komposition P_{12} gilt:*

1. $ET_{12} = \text{cont}(\text{prune}((ET_1 \| EL_2) \cup (EL_1 \| ET_2)))$,
2. $EL_{12} = (EL_1 \| EL_2) \cup ET_{12}$.

Beweis.
1. „$\subseteq$“:
Da beide Seiten der Gleichung unter der Fortsetzung cont abgeschlossen sind, genügt es ein präfix-minimales Element w von ET_{12} zu betrachten. Diese Element ist aufgrund der Definition der Menge der Fehler-Traces in MIT_{12} oder in $PrET_{12}$ enthalten.

- Fall 1 ($w \in MIT_{12}$): Aus der Definition von MIT folgt, dass es eine Aufteilung $w = xa$ gibt mit $(p_{01}, p_{02}) \overset{x}{\Rightarrow}_{12} (p_1, p_2) \wedge a \in I_{12} \wedge (p_1, p_2) \overset{a}{\not\rightarrow}_{12}$. Da $I_{12} = (I_1 \cup I_2) \backslash (O_1 \cup O_2)$ ist, folgt $a \in (I_1 \cup I_2)$ und $a \notin (O_1 \cup O_2)$. Es wird unterschieden, ob $a \in (I_1 \cap I_2)$ oder $a \in (I_1 \cup I_2) \backslash (I_1 \cap I_2)$ ist.
  - Fall 1a) ($a \in (I_1 \cap I_2)$): Durch Projektion des Ablaufes auf die einzelnen Transitionssysteme erhält man oBdA $p_{01} \overset{x_1}{\Rightarrow}_1 p_1 \overset{a}{\not\rightarrow}_1$ und $p_{02} \overset{x_2}{\Rightarrow}_2 p_2 \overset{a}{\not\rightarrow}_2$ oder $p_{02} \overset{x_2}{\Rightarrow}_2 p_2 \overset{a}{\rightarrow}_2$ mit $x \in x_1 \| x_2$. Daraus kann $x_1 a \in \text{cont}(MIT_1) \subseteq ET_1$ und $x_2 a \in EL_2$ ($x_2 a \in MIT_2$ oder $x_2 a \in L_2$) gefolgert werden. Damit folgt $w \in (x_1 \| x_2) \cdot \{a\} \subseteq (x_1 a) \| (x_2 a) \subseteq ET_1 \| EL_2$, und somit ist w in der rechten Seite der Gleichung enthalten.
  - Fall 1b) ($a \in (I_1 \cup I_2) \backslash (I_1 \cap I_2)$): OBdA gilt $a \in I_1$. Durch die Projektion auf die einzelnen Komponenten erhält man: $p_{01} \overset{x_1}{\Rightarrow}_1 p_1 \overset{a}{\not\rightarrow}_1$ und $p_{02} \overset{x_2}{\Rightarrow}_2 p_2$ mit $x \in x_1 \| x_2$. Daraus folgt $x_1 a \in \text{cont}(MIT_1) \subseteq ET_1$ und $x_2 \in L_2 \subseteq EL_2$.

Somit gilt $w \in (x_1 \| x_2) \cdot \{a\} \subseteq (x_1 a) \| x_2 \subseteq ET_1 \| EL_2$. Dies ist eine Teilmenge der rechten Seite der Gleichung.

- Fall 2 ($w \in PrET_{12}$): Aus der Definition von $PrET$ und prune folgt, dass ein $v \in O_{12}^*$ gibt, so dass $(p_{01}, p_{02}) \overset{w}{\Rightarrow}_{12} (p_1, p_2) \overset{v}{\Rightarrow}_{12} (p_1', p_2')$ gilt mit $(p_1', p_2') \in E_{12}$ und $w = \text{prune}(wv)$. Durch Projektion auf die Komponenten erhält man $p_{01} \overset{w_1}{\Rightarrow}_1 p_1 \overset{v_1}{\Rightarrow}_1 p_1'$ und $p_{02} \overset{w_2}{\Rightarrow}_2 p_2 \overset{v_2}{\Rightarrow}_2 p_2'$ mit $w \in w_1 \| w_2$ und $v \in v_1 \| v_2$. Aus $(p_1', p_2') \in ET_{12}$ folgt, dass es sich entweder um einen geerbten oder einen neuen Fehler handelt. Bei einem geerbten wäre bereits einer der beiden Zustände p_1' bzw. p_2' ein Fehler-Zustand gewesen. Ein neuer Fehler hingegen wäre durch das fehlen fehlende Sicherstellen der Synchronisation (fehlende must-Transition) in einer der Komponenten entstanden.

  - Fall 2a) (geerbter Fehler): OBdA gilt $p_1' \in E_1$. Daraus folgt, $w_1 v_1 \in StET_1 \subseteq \text{cont}(PrET_1) \subseteq ET_1$. Da $p_{02} \overset{w_2 v_2}{\Rightarrow}_2$ gilt, erhält man $w_2 v_2 \in L_2 \subseteq EL_2$. Dadurch ergibt sich $wv \in ET_1 \| EL_2$ mit $w = \text{prune}(wv)$ und somit ist w in der rechten Seite der Gleichung enthalten.

  - Fall 2b) (neuer Fehler): OBdA gilt $a \in I_1 \cap O_2$ mit $p_1' \overset{a}{\not\rightarrow}_1$ und $p_2' \overset{a}{\dashrightarrow}_2$. Daraus folgt $w_1 v_1 a \in MIT_1 \subseteq ET_1$ und $w_2 v_2 a \in L_2 \subseteq EL_2$. Damit ergibt sich $wva \in ET_1 \| EL_2$, da $a \in O_1 \subseteq O_{12}$ gilt $w = \text{prune}(wva)$ und somit ist w in der rechten Seite der Gleichung enthalten.

1. „$\supseteq$“:
Wegen der Abgeschlossenheit beider Seiten der Gleichung gegenüber cont wird auch in diesem Fall nur ein präfix-minimales Element $x \in \text{prune}((ET_1 \| EL_2) \cup (EL_1 \| ET_2))$ betrachtet. Da x durch die Anwendung der prune-Funktion entstanden ist, existiert ein $y \in O_{12}^*$ mit $xy \in (ET_1 \| EL_2) \cup (EL_1 \| ET_2)$. OBdA wird davon ausgegangen, dass $xy \in ET_1 \| EL_2$ gilt, d.h. es gibt $w_1 \in ET_1$ und $w_2 \in EL_2$ mit $xy \in w_1 \| w_2$.
Im Folgenden wird für alle Fälle von xy gezeigt, dass es ein $v \in PrET_{12} \cup MIT_{12}$ gibt, das ein Präfix von xy ist und v entweder auf einen Input I_{12} endet oder $v = \varepsilon$. Damit muss v ein Präfix von x sein. ε ist Präfix von jedem Wort und sobald v mindestens einen Buchstaben enthält, muss das Ende von v vor dem Anfang von $y \in O_{12}^*$ liegen. Dadurch ist ein Präfix von x in $PrET_{12} \cup MIT_{12}$ enthalten und somit gilt $x \in ET_{12}$, da ET die Fortsetzung der Mengenvereinigung aus $PrET$ und MIT ist.
Sei v_1 das kürzeste Präfix von w_1 in $PrET_1 \cup MIT_1$. Falls $w_2 \in L_2$, so sei $v_2 = w_2$, sonst soll v_2 das kürzeste Präfix von w_2 in $PrET_2 \cup MIT_2$ sein. Jede Aktion in v_1 und v_2 hängt mit einer aus xy zusammen. Es kann nun davon ausgegangen werden, dass entweder $v_2 = w_2 \in L_2$ gilt oder die letzte Aktion von v_1 vor oder gleichzeitig mit der letzten Aktion von v_2 statt findet. Ansonsten endet $v_2 \in PrET_2 \cup MIT_2$ vor v_1 und somit ist dieser Fall analog zu v_1 endet vor v_2.

- Fall 1 ($v_1 = \varepsilon$): Da $\varepsilon \in PrET_1 \cup MIT_1$, ist bereits in P_1 ein Fehler-Zustand lokal erreichbar. $\varepsilon \in MIT_1$ ist nicht möglich, da jedes Element aus MIT nach Definition mindestens die Länge 1 haben muss. Mit der Wahl $v_2' = v' = \varepsilon$ ist v_2' ein Präfix von v_2.

- Fall 2 ($v_1 \neq \varepsilon$): Aufgrund der Definitionen von $PrET$ und MIT endet v_1 auf ein $a \in I_1$, d.h. $v_1 = v_1'a$. v' sei das Präfix von xy, das mit der letzten Aktion von v_1 endet, d.h. mit a und $v_2' = v'|_{\Sigma_2}$. Falls $v_2 = w_2 \in L_2$, dann ist v_2' ein Präfix von v_2. Falls $v_2 \in PrET_2 \cup MIT_2$ gilt, dann ist durch die Annahme, dass v_2 nicht vor v_1 endet, v_2' ein Präfix von v_2. Im Fall $v_2 \in MIT_2$ weiß man zusätzlich, dass v_2 auf $b \in I_2$ endet. Es kann jedoch $a = b$ gelten.

In allen Fällen erhält man $v_2' = v'|_{\Sigma_2}$ ist ein Präfix von v_2 und $v' \in v_1 \| v_2'$ ist ein Präfix von xy. Es kann nur für die Fälle $a \notin I_2$ gefolgert werden, dass $p_{02} \overset{v_2'}{\Longrightarrow}_2$ gilt.

- Fall I ($v_1 \in MIT_1$ und $v_1 \neq \varepsilon$): Es gibt einen Ablauf der Form $p_{01} \overset{v_1'}{=\!\Rightarrow}_1 p_1 \overset{a}{\not\longrightarrow}_1$ und es gilt $v' = v''a$.

  - Fall Ia) ($a \notin \Sigma_2$): Es gilt $p_{02} \overset{v_2'}{=\!\Rightarrow}_2 p_2$ mit $v'' \in v_1' \| v_2'$. Dadurch erhält man $(p_{01}, p_{02}) \overset{v''}{=\!\Rightarrow}_{12} (p_1, p_2) \overset{a}{\not\longrightarrow}_{12}$ mit $a \in I_{12}$. Somit wird $v := v''a = v' \in MIT_{12}$ gewählt.

  - Fall Ib) ($a \in I_2$ und $v_2' \in MIT_2$): Es gilt $v_2' = v_2''a$ mit $p_{02} \overset{v_2''}{=\!\Rightarrow}_2 \overset{a}{\not\longrightarrow}_2$ und $v'' \in v_1' \| v_2''$. a ist für P_2, ebenso wie für P_1, ein nicht sichergestellter Input. Daraus folgt, dass $(p_1, p_2) \overset{a}{\not\longrightarrow}_{12}$ gilt. Es wird ebenfalls $v := v''a = v' \in MIT_{12}$ gewählt.

  - Fall Ic) ($a \in I_2$ und $v_2' \in L_2 \backslash MIT_2$): Es gilt $p_{02} \overset{v_2''}{=\!\Rightarrow}_2 p_2 \overset{a}{\longrightarrow}_2$ mit $v_2' = v_2''a$. Da die gemeinsamen Inputs synchronisiert werden, folgt $(p_1, p_2) \overset{a}{\not\longrightarrow}_{12}$ bereits aus $q_1 \overset{a}{\not\longrightarrow}_1$. Somit kann hier nochmals $v := v''a = v' \in MIT_{12}$ gewählt werden.

  - Fall Id) ($a \in O_2$): Es gilt $v_2' = v_2''a$ und $p_{02} \overset{v_2'}{=\!\Rightarrow}_2$. Man erhält also $p_{02} \overset{v_2''}{=\!\Rightarrow}_2 p_2 \overset{a}{\dashrightarrow}_2$ mit $v'' \in v_1' \| v_2''$. Daraus ergibt sich $(p_{01}, p_{02}) \overset{v''}{=\!\Rightarrow}_{12} (p_1, p_2)$ mit $p_2 \overset{a}{\dashrightarrow}_1$, $p_1 \overset{a}{\not\longrightarrow}_1$, $a \in I_1$ und $a \in O_2$, somit gilt $(p_1, p_2) \in E_{12}$. Es wird $v := \text{prune}(v'') \in PrET_{12}$ gewählt.

- Fall II ($v_1 \in PrET_1$): $\exists u_1 \in O_1^* : p_{01} \overset{v_1}{=\!\Rightarrow}_1 p_1 \overset{u_1}{=\!\Rightarrow}_1 p_1'$ mit $p_1' \in E_1$. Im Fall $v_1' \neq \varepsilon$ kann das a, auf das v_1 endet, ebenfalls der letzte Buchstabe von v_2 sein. Im Fall von $v_2 \in MIT_2$ kann somit $a = b$ gelten, wodurch $v_2 = v_2'$ gilt. Dieser Fall verläuft jedoch analog zu Fall Ic) und wird hier nicht weiter betrachtet. Es gilt für alle anderen Fälle $p_{02} \overset{v_2'}{=\!\Rightarrow}_2 p_2$ mit $(p_{01}, p_{02}) \overset{v'}{=\!\Rightarrow}_{12} (p_1, p_2)$.

  - Fall IIa) $\Big(u_2 \in (O_1 \cap I_2)^*, c \in (O_1 \cap I_2)$, sodass u_2c Präfix von $u_1|_{I_2}$ mit $p_2 \overset{u_2}{\Longrightarrow}_2 p_2' \overset{c}{\not\longrightarrow}_2\Big)$: Für das Präfix $u_1'c$ von u_1 mit $(u_1'c)|_{I_2} = u_2c$ weiß man, dass $q_1 \overset{u_1'}{=\!\Rightarrow}_1 q_1'' \overset{c}{\dashrightarrow}_1$. Somit gilt $u_1' \in u_1' \| u_2$ und $(p_1, p_2) \overset{u_1'}{=\!\Rightarrow}_{12} (q_1'', q_2') \in E_{12}$, da für P_2 der entsprechende Input nicht sichergestellt wird, der mit dem c Output von P_1 zu koppeln wäre. Es handelt sich also um einen neuen Fehler.

Es wird $v := \text{prune}(v'u_1') \in PrET_{12}$ gewählt, dies ist ein Präfix von v', da $u_1 \in O_1^*$.

– Fall IIb) $\left(p_2 \overset{u_2}{\Longrightarrow}_2 p_2' \text{ mit } u_2 = u_1|_{I_2}\right)$: Es gilt $u_1 \in u_1 \| u_2$ und $(p_1, p_2) \overset{u_1}{\Rightarrow}_{12} (p_1', p_2') \in E_{12}$, da $p_1' \in E_1$ und somit handelt es sich in P_{12} um einen geerbten Fehler. Nun wird $v := \text{prune}(v'u_1) \in PrET_{12}$ gewählt, das wiederum ein Präfix von v' ist.

2.:
Durch die Definitionen ist klar, dass $L_i \subseteq EL_i$ und $ET_i \subseteq EL_i$ gilt. Die Argumentation startet auf den rechten Seite der Gleichung:

$$\begin{aligned}
(EL_1 \| EL_2) \cup ET_{12} &\overset{3.5}{=} ((L_1 \cup ET_1) \parallel (L_2 \cup ET_2)) \cup ET_{12} \\
&= (L_1 \| L_2) \cup \underbrace{(L_1 \| ET_2)}_{\substack{\subseteq (EL_1 \| ET_2) \\ \overset{1.}{\subseteq} ET_{12}}} \cup \underbrace{(ET_1 \| L_2)}_{\substack{\subseteq (ET_1 \| EL_2) \\ \overset{1.}{\subseteq} ET_{12}}} \cup \underbrace{(ET_1 \| ET_2)}_{\substack{\subseteq (EL_1 \| ET_2) \\ \overset{1.}{\subseteq} ET_{12}}} \cup ET_{12} \\
&= (L_1 \| L_2) \cup ET_{12} \\
&\overset{2.2}{=} L_{12} \cup ET_{12} \\
&\overset{3.5}{=} EL_{12}.
\end{aligned}$$

□

Korollar 3.8 (*Kommunikationsfehler-Präkongruenz*). *Die Relation $\sqsubseteq_E$ ist eine Präkongruenz bezüglich $\cdot \| \cdot$.*

Beweis. Es muss gezeigt werden: Wenn $P_1 \sqsubseteq_E P_2$ gilt, dann für jedes komponierbare P_3 auch $P_{31} \sqsubseteq_E P_{32}$. D.h. es ist zu zeigen, dass aus $ET_1 \subseteq ET_2$ und $EL_1 \subseteq EL_2$, $ET_{31} \subseteq ET_{32}$ und $EL_{31} \subseteq EL_{32}$ folgt. Dies ergibt sich aus der Monotonie von cont, prune und $\cdot \| \cdot$ auf Sprachen wie folgt:

- $$\begin{aligned} ET_{31} &\overset{3.7\ 1.}{=} \text{cont}\left(\text{prune}\left((ET_3 \| EL_1) \cup (EL_3 \| ET_1)\right)\right) \\ &\overset{\substack{ET_1 \subseteq ET_2 \\ \text{und} \\ EL_1 \subseteq EL_2}}{\subseteq} \text{cont}\left(\text{prune}\left((ET_3 \| EL_2) \cup (EL_3 \| ET_2)\right)\right) \\ &\overset{3.7\ 1.}{=} ET_{32}, \end{aligned}$$
- $$\begin{aligned} EL_{31} &\overset{3.7\ 2.}{=} (EL_3 \| EL_1) \cup E_{31} \\ &\overset{\substack{EL_1 \subseteq EL_2 \\ \text{und} \\ ET_{31} \subseteq ET_{32}}}{\subseteq} (EL_3 \| EL_2) \cup ET_{32} \\ &\overset{3.7\ 2.}{=} EL_{32}. \end{aligned}$$

□

Lemma 3.9 (*Verfeinerung mit Kommunikationsfehlern*). *Gegeben sind zwei MEIOs P_1 und P_2 mit der gleichen Signatur. Wenn $U\|P_1 \sqsubseteq_E^{\mathrm{B}} U\|P_2$ für alle Partner U gilt, dann folgt daraus die Gültigkeit von $P_1 \sqsubseteq_E P_2$.*

Beweis. Da P_1 und P_2 die gleiche Signaturen haben wird $I := I_1 = I_2$ und $O := O_1 = O_2$ definiert. Für jeden Partner U gilt $I_U = O$ und $O_U = I$.
Um $P_1 \sqsubseteq_E P_2$ zu zeigen, wird nachgeprüft, ob folgendes gilt:

- $ET_1 \subseteq ET_2$,
- $EL_1 \subseteq EL_2$.

Für ein gewähltes präfix-minimales Element $w \in ET_1$ wir gezeigt, dass dieses w oder eines seiner Präfixe in ET_2 enthalten ist. Dies ist möglich, da die beiden Mengen ET_1 und ET_2 durch cont abgeschlossen sind.

- Fall 1 ($w = \varepsilon$): Es handelt sich um einen lokal erreichbaren Fehler-Zustand in P_1. Für U wird ein Transitionssystem verwendet, das nur aus dem Startzustand und einer must-Schleife für alle Inputs $x \in I_U$ besteht. Somit kann P_1 die im Prinzip gleichen Fehler-Zustände lokal erreichen wie alle möglichen as-Implementierungen von $U\|P_1$ zusammen. Daraus folgt, dass auch mindestens eine as-Implementierung von $U\|P_2$ einen lokal erreichbaren Fehler-Zustand haben muss. Durch die Definition von U kann dieser Fehler nur von P_2 geerbt sein. Es muss also in P_2 ein Fehler-Zustand durch interne Aktionen und Outputs erreichbar sein, d.h. es gilt $\varepsilon \in PrET_2$.
- Fall 2 ($w = x_1 \dots x_n x_{n+1} \in \Sigma^+$ mit $n \geq 0$ und $x_{n+1} \in I = O_U$): Es wird der folgende Partner U betrachtet (siehe auch Abbildung 3.1):
  - $U = \{p_0, p_1, \dots, p_{n+1}\}$,
  - $p_{0U} = p_0$,
  - $\longrightarrow_U = \{(p_j, x_{j+1}, p_{j+1}) \mid 0 \leq j \leq n\}$
    $\cup \{(p_j, x, p_{n+1}) \mid x \in I_U \backslash \{x_{j+1}\}, 0 \leq j \leq n\}$
    $\cup \{(p_{n+1}, x, p_{n+1}) \mid x \in I_U\}$,
  - $E_U = \emptyset$.

  Für w können nun zwei Fälle unterschieden werden. Aus beiden wird folgen, dass für mindestens eine as-Implementierung P' von $U\|P_1$ $\varepsilon \in PrET(P')$ gilt.
  - Fall 2a) ($w \in MIT_1$): In $U\|P_1$ erhält man $(p_0, p_{01}) \stackrel{x_1 \dots x_n}{\Longrightarrow}_{U\|P_1} (p_n, p')$ mit $p' \stackrel{x_{n+1}}{\not\longrightarrow}_1$ und $p_n \stackrel{x_{n+1}}{\longrightarrow}_U$. Deshalb gilt $(p_n, p') \in E_{U\|P_1}$. Da alle Aktionen aus w bis auf x_{n+1} synchronisiert werden und $I \cap I_U = \emptyset$, gilt $x_1, \dots, x_n \in O_{U\|P_1}$. Da $(p_n, p') \in E_{U\|P_1}$ gibt es mindestens ein P' in as-impl($U\|P_1$), das diesen Fehler-Zustand ebenfalls enthält. Daraus ergibt sich dann $\varepsilon \in PrET(P')$.

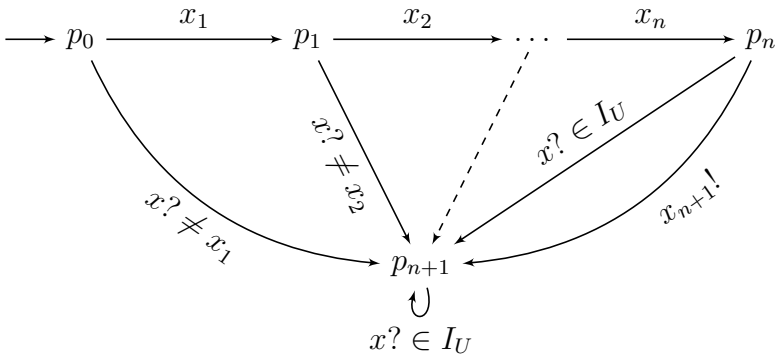

Abbildung 3.1: $x? \neq x_j$ steht für alle $x \in I_U \backslash \{x_j\}$

– Fall 2b) ($w \in PrET_1$): In der Parallelkomposition von U und P_1 erhält man $(p_0, p_{01}) \overset{w}{\Longrightarrow}_{U\|P_1} (p_{n+1}, p'') \overset{u}{\Longrightarrow}_{U\|P_1} (p_{n+1}, p')$ für $u \in O^*$ und $p' \in E_1$. Daraus folgt $(p_{n+1}, p') \in E_{U\|P_1}$ und somit $wu \in StET(U\|P_1)$. Da alle Aktionen in w synchronisiert werden und $I \cap I_U = \emptyset$, gilt $x_1, \dots, x_n, x_{n+1} \in O_{U\|P_1}$ und, da $u \in O^*$, folgt $u \in O^*_{U\|P_1}$. Somit ergibt sich für eine as-Implementierung P' von $U\|P_1$ $\varepsilon \in PrET(P')$.

Da $\varepsilon \in PrET(P')$ für ein P' aus as-impl$(U\|P_1)$ gilt, kann durch $U\|P_1 \sqsubseteq^{\mathrm{B}}_E U\|P_2$ geschlossen werden, dass auch in mindestens einer as-Implementierung von $U\|P_2$ ein Fehler-Zustand lokal erreichbar sein muss. Da as-Implementierungen die Definition 1.3 erfüllen müssen, muss falls in einer as-Implementierung von $U\|P_2$ ein Fehler lokal erreichbare ist auch in $U\|P_2$ ein Fehler-Zustand lokal erreichbar sein. Der in $U\|P_2$ erreichbare Fehler kann geerbt oder neu sein.

– Fall 2i) (neuer Fehler): Da jeder Zustand von U alle Inputs $x \in O = I_U$ durch must-Transitionen sicherstellt, muss ein lokal erreichbarer Fehler-Zustand der Form sein, dass ein Output $a \in O_U$ von U möglich ist, der nicht mit einem passenden Input aus P_2 synchronisiert werden muss (P_2 enthält die entsprechende a Transitionen nicht als must-Transition). Durch die Konstruktion von U sind in p_{n+1} keine Outputs möglich. Ein neuer Fehler muss also die Form (p_i, p') haben mit $i \leq n, p' \overset{x_{i+1}}{\not\longrightarrow}_2$ und $x_{i+1} \in O_U = I$. Durch Projektion erhält man dann $p_{02} \overset{x_1 \dots x_i}{\Longrightarrow}_2 p' \overset{x_{i+1}}{\not\longrightarrow}_2$ und damit gilt $x_1 \dots x_{i+1} \in MIT_2 \subseteq ET_2$. Somit ist ein Präfix von w in ET_2 enthalten.

– Fall 2ii) (geerbter Fehler): U hat $x_1 \dots x_i u$ mit $u \in I^*_U = O^*$ ausgeführt und ebenso hat P_2 dieses Wort abgearbeitet. Durch dies hat P_2 einen Zustand in E_2 erreicht, da von U keine Fehler geerbt werden können. Es gilt dann prune$(x_1 \dots x_i u)$ = prune$(x_1 \dots x_i) \in PrET_2 \subseteq ET_2$. Da $x_1 \dots x_i$ ein Präfix von w ist, führt in diesem Fall eine Verlängerung um lokale Aktionen von einem Präfix von w zu einem Fehler-Zustand. Da ET der Menge aller Verlängerungen von gekürzten Fehler-Traces entspricht, ist $x_1 \dots x_i$ in ET_2 enthalten und somit ist ein Präfix von w in ET_2 enthalten.

Um die andere Inklusion zu beweisen, reicht es aufgrund der ersten Inklusion und der Definition von EL aus zu zeigen, dass $L_1 \backslash ET_1 \subseteq EL_2$ gilt.
Es wird dafür ein beliebiges $w \in L_1 \backslash ET_1$ gewählt und gezeigt, dass es in EL_2 enthalten ist.

- Fall 1 ($w = \varepsilon$): Da ε immer in EL_2 enthalten ist, muss hier nichts gezeigt werden.
- Fall 2 ($w = x_1 \dots x_n$ mit $n \geq 1$): Es wird ein Partner U wie folgt konstruiert (siehe dazu auch Abbildung 3.2):
  - $U = \{p_0, p_1, \dots, p_n, p\}$,
  - $p_{0U} = p_0$,
  - $\longrightarrow_U = \{(p_j, x_{j+1}, p_{j+1}) \mid 0 \leq j < n\}$
    $\cup \{(p_j, x, p) \mid x \in I_U \backslash \{x_{j+1}\}, 0 \leq j < n\}$
    $\cup \{(p, x, p) \mid x \in I_U\}$,
  - $E_U = \{p_n\}$.

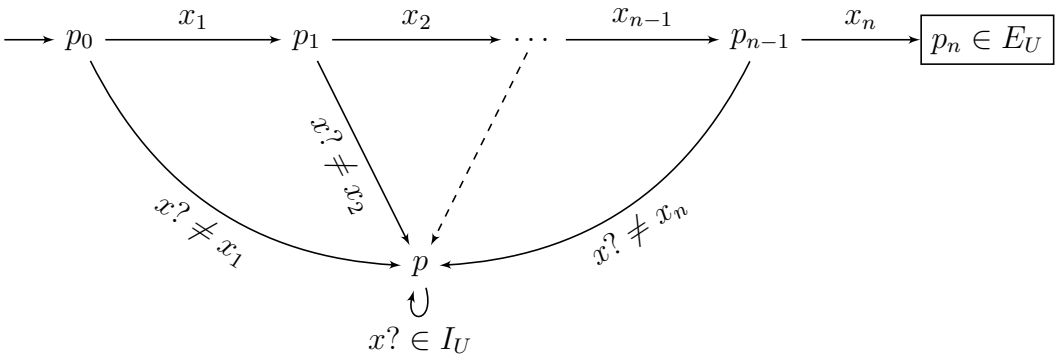

Abbildung 3.2: $x? \neq x_j$ steht für alle $x \in I_U \backslash \{x_j\}$, p_n ist der einzige Fehler-Zustand

Da $p_{01} \overset{w}{\Rightarrow}_1 p'$ gilt, kann man schließen, dass $U \| P_1$ einen lokal erreichbaren geerbten Fehler hat. Es gibt also auch mindestens eine as-Implementierung in der Menge as-impl$(U \| P_1)$, die diesen lokal erreichbaren Fehler implementiert. Somit muss es eine as-Implementierung von $U \| P_2$ geben, die ebenfalls einen lokal erreichbaren Fehler-Zustand hat. Aufgrund von Definition 1.3 3. muss dann ebenfalls ein Fehler-Zustand in $U \| P_2$ lokal erreichbar sein.

- Fall 2a) (neuer Fehler aufgrund von $x_i \in O_U$ und $p_{02} \overset{x_1 \dots x_{i-1}}{\Longrightarrow}_2 p'' \overset{x_i}{\not\longrightarrow}_2$): Es gilt $x_1 \dots x_i \in MIT_2$ und somit $w \in EL_2$. Anzumerken ist, dass es nur auf diesem Weg Outputs von U möglich sind, deshalb gibt es keine anderen Outputs von U, die zu einem neuen Fehler führen können.
- Fall 2b) (neuer Fehler aufgrund von $a \in O = I_U$): Der einzige Zustand, in dem U nicht alle Inputs erlaubt sind, ist p_n, der bereits ein Fehler-Zustand ist. Da

in diesem Fall dieser Zustand in $U\|P_2$ erreichbar ist, besitzt das komponierte MEIO einen geerbten Fehler und es gilt $w \in L_2 \subseteq EL_2$, wegen dem folgenden Fall 2c).

- Fall 2c) (geerbter Fehler von U): Da p_n der einzige Fehler-Zustand in U ist und alle Aktionen synchronisiert sind, ist dies nur möglich, wenn $p_{02} \stackrel{x_1 \ldots x_n}{=\!=\!\Rightarrow}_2$ gilt. In diesem Fall gilt $w \in L_2 \subseteq EL_2$.
- Fall 2d) (geerbter Fehler von P_2): Es gilt dann $p_{02} \stackrel{x_1 \ldots x_i u}{=\!=\!\Rightarrow}_2 p' \in E_2$ für $i \geq 0$ und $u \in O^*$. Somit ist $x_1 \ldots x_i u \in StET_2$ und damit $\text{prune}(x_1 \ldots x_i u) = \text{prune}(x_1 \ldots x_i) \in PrET_2 \subseteq EL_2$. Es gilt also $w \in EL_2$.

□

Der folgende Satz sagt aus, dass $\sqsubseteq_E$ die gröbste Präkongruenz ist, die charakterisiert werden soll, also gleich der vollständig abstrakten Präkongruenz $\sqsubseteq_E^{\mathrm{C}}$.

Satz 3.10 (*Vollständige Abstraktheit für Kommunikationsfehler-Semantik*). *Für zwei MEIOs P_1 und P_2 mit derselben Signatur gilt $P_1 \sqsubseteq_E^{\mathrm{C}} P_2 \Leftrightarrow P_1 \sqsubseteq_E P_2$.*

Beweis.
„⇐“: Nach Definition gilt, genau dann wenn $\varepsilon \in ET(P)$, ist ein Fehler-Zustand lokal erreichbar in P. $P_1 \sqsubseteq_E P_2$ impliziert, dass $\varepsilon \in ET_2$ gilt, wenn $\varepsilon \in ET_1$. Somit ist ein Fehler-Zustand in P_1 nur dann lokal erreichbar, wenn dieser auch in P_2 lokal erreichbar ist. Falls es also eine as-Implementierung von P_1 gibt, in der ein Fehler-Zustand lokal erreichbar ist, dann gibt es auch mindestens eine as-Implementierung von P_2, die einen Fehler-Zustand lokal erreichen kann. Dadurch folgt, dass $P_1 \sqsubseteq_E^{\mathrm{B}} P_2$ gilt, da $\sqsubseteq_E^{\mathrm{B}}$ in Definition 3.2 über die lokale Erreichbarkeit der Fehler-Zustande in den as-Implementierungen definiert wurde und die ET-Mengen von P_1 und P_2 auch in der Vereinigung der Traces ihrer as-Implementierungen enthalten sind, wie in Proposition 3.6, ausgedrückt werden können. Es ist also $\sqsubseteq_E$ in $\sqsubseteq_E^{\mathrm{B}}$ enthalten. Wie in Korollar 3.8 gezeigt, ist $\sqsubseteq_E$ eine Präkongruenz bezüglich $\cdot\|\cdot$. Da $\sqsubseteq_E^{\mathrm{C}}$ die gröbste Präkongruenz bezüglich $\cdot\|\cdot$ ist, die in $\sqsubseteq_E^{\mathrm{B}}$ enthalten ist, muss $\sqsubseteq_E$ in $\sqsubseteq_E^{\mathrm{C}}$ enthalten sein. Es folgt also aus $P_1 \sqsubseteq_E P_2$, dass auch $P_1 \sqsubseteq_E^{\mathrm{C}} P_2$ gilt.

„⇒“: Durch die Definition von $\sqsubseteq_E^{\mathrm{C}}$ als Präkongruenz in 3.2 folgt aus $P_1 \sqsubseteq_E^{\mathrm{C}} P_2$, dass $U\|P_1 \sqsubseteq_E^{\mathrm{C}} U\|P_2$ für alle MEIOs U gilt, die mit P_1 komponierbar sind. Da $\sqsubseteq_E^{\mathrm{C}}$ nach Definition auch in $\sqsubseteq_E^{\mathrm{B}}$ enthalten sein soll, folgt aus $U\|P_1 \sqsubseteq_E^{\mathrm{C}} U\|P_2$ auch die Gültigkeit von $U\|P_1 \sqsubseteq_E^{\mathrm{B}} U\|P_2$ für alle diese MEIOs U. Mit Lemma 3.9 folgt dann $P_1 \sqsubseteq_E P_2$. □

Es wurde somit eine Kette an Folgerungen gezeigt, die sich zu einem Ring schließt. Dies ist in Abbildung 3.3 dargestellt.

Angenommen man definiert, dass P_1 P_2 verfeinern soll, genau dann wenn für alle Partner MEIOs U, für die P_2 fehler-frei mit U kommuniziert, folgt, dass P_1 ebenfalls fehlerfrei mit U kommuniziert. Dann wird auch diese Verfeinerung durch $\sqsubseteq_E$ charakterisiert.

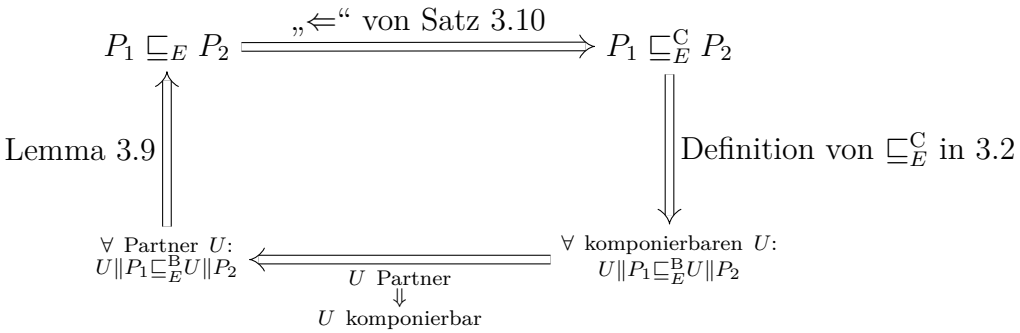

Abbildung 3.3: Folgerungskette der Fehler-Relationen

Korollar 3.11. *Es gilt:* $P_1 \sqsubseteq_E P_2 \Leftrightarrow U \| P_1 \sqsubseteq_E^{\mathrm{B}} U \| P_2$ *für alle Partner* U*.*

3.2 Testing-Ansatz

Der Testing-Ansatz stützt sich auf den Ansatz, der in [BV15b] angewendet wurde. Jedoch sind Tests dort Tupel aus einer Implementierung und einer Menge an Aktionen, über denen synchronisiert werden soll. Die MEIOs, die hier komponiert werden bringen die Menge Synch an Aktionen, die synchronisiert werden bereits mit. Es wird im Gegensatz zu [BV15b] mit Inputs und Outputs gearbeitet und dadurch scheint der Ansatz die gemeinsamen Aktionen zu synchronisieren natürlicher, wie eine Menge beim Test vorzugeben.
Die Definition von lokaler Erreichbarkeit eines Fehler-Zustandes soll aus dem Erweiterungs-Ansatz übernommen werden.

Definition 3.12 (*Test und Verfeinerung*). *Ein* Test T *ist eine Implementierung. Ein MEIO* P as-erfüllt *einen Test* T*, falls* $S \| T$ *fehler-frei ist für alle* $S \in \text{as-impl}(P)$*. Es wird dann* $P \, sat_{\text{as}} \, T$ *geschrieben.*
Ein MEIO P verfeinert P'*, falls für alle Tests* T*:* $P' \, sat_{\text{as}} \, T \Rightarrow P \, sat_{\text{as}} \, T$*.*

Die Definition 3.3 der Kommunikationsfehler-Traces eines MEIOs P kann aus dem Erweiterungs-Ansatz übernommen werden. Ebenso wie die Definition der Kommunikationsfehler-Semantik aus 3.5. Daraus kann dann auch in diesem Ansatz der Satz 3.7, das Korollar 3.8 und die Propositionen 3.4 und 3.6 beweisen werden.
Die Basisrelation aus dem Erweiterungs-Ansatz gibt es in diesem Ansatz nicht, somit kann das Lemma 3.9 nicht in dieser Art formuliert werden. Jedoch kann hier jetzt mit Tests gearbeitet werden.

Lemma 3.13 (*Testing-Verfeinerung mit Kommunikationsfehlern*). *Gegeben sind zwei MEIOs* P_1 *und* P_2 *mit der gleichen Signatur. Wenn für alle Tests* T*, die*

Partner von P_1 bzw. P_2 sind, $P_2\, sat_{\text{as}}\, T \Rightarrow P_1\, sat_{\text{as}}\, T$ gilt, dann folgt daraus die Gültigkeit von $P_1 \sqsubseteq_E P_2$.

Beweis. Da P_1 und P_2 die gleichen Signaturen haben wird $I := I_1 = I_2$ und $O := O_1 = O_2$ definiert. Für jeden Test Partner T gilt $I_T = O$ und $O_T = I$.
Um $P_1 \sqsubseteq_E P_2$ zu zeigen, wird nachgeprüft, ob folgendes gilt:

- $ET_1 \subseteq ET_2$,
- $EL_1 \subseteq EL_2$.

Für ein gewähltes präfix-minimales Element $w \in ET_1$ wird gezeigt, dass dies w oder eines seiner Präfixe in ET_2 enthalten ist. Dies ist möglich, da die beiden Mengen ET_1 und ET_2 durch cont abgeschlossen sind.
Mit Proposition 3.6 folgt aus $w \in ET_1$, dass es auch eine as-Implementierung P_1' von P_1 geben muss, für die w ebenfalls in $ET_{P_1'}$ enthalten ist.

- Fall 1 ($w = \varepsilon$): Es ist ein Fehler-Zustand in P_1' lokal erreichbar. Für T wird ein Transitionssystem verwendet, das nur aus dem Startzustand und einer must-Schleife für alle Inputs $x \in I_T$ besteht. Somit kann P_1' die im Prinzip gleichen Fehler-Zustände lokal erreichen wie $P_1' \| T$. P_1' ist in Parallelkomposition mit T nicht fehler-frei somit gilt $P_1\, sat_{\text{as}}\, T$ nicht. Es darf also auch P_2 den Test T nicht as-erfüllen, wegen der Implikation $P_2\, sat_{\text{as}}\, T \Rightarrow P_1\, sat_{\text{as}}\, T$. Damit P_2 T nicht as-erfüllt muss es eine as-Implementierungen P_2' geben, die in Parallelkomposition mit T zu einem nicht fehler-freien System führt. Da T ein Partner von P_2' ist, gibt es in der Parallelkomposition nur lokale Aktionen. Somit muss in $P_2' \| T$ ein Fehler lokal erreichbar sein. Durch die Definition von T kann dieser Fehler nur von P_2' geerbt sein. In P_2' kann dieser Fehler-Zustand nur durch interen Aktionen und Outputs erreichbar sein, da T keine Outputs besitzt, die man mit Inputs aus P_2' synchronisieren könnte und unsynchronisierte Aktionen sind in einer Parallelkomposition von Partner nicht möglich. Somit gilt $\varepsilon \in PrET_{P_2'} \subseteq ET_{P_2'}$. Mit Proposition 3.6 folgt daraus $\varepsilon \in ET_2$.
- Fall 2 ($w = x_1 \ldots x_n x_{n+1} \in \Sigma^+$ mit $n \geq 0$ und $x_{n+1} \in I = O_T$): Es wird der folgende Partner T betrachtet (dieser entspricht bis auf die Benennung der Mengen dem Transitionssystem U aus Abbildung 3.1):
  - $T = \{p_0, p_1, \ldots, p_{n+1}\}$,
  - $p_{0T} = p_0$,
  - $\longrightarrow_T = \{(p_j, x_{j+1}, p_{j+1}) \mid 0 \leq j \leq n\}$
    $\cup \{(p_j, x, p_{n+1}) \mid x \in I_T \backslash \{x_{j+1}\}, 0 \leq j \leq n\}$
    $\cup \{(p_{n+1}, x, p_{n+1}) \mid x \in I_T\}$,
  - $E_T = \emptyset$.

Für w können nun zwei Fälle unterschieden werden, für die beide $\varepsilon \in PrET(P_1'\|T)$ folgen wird.

- Fall 2a) ($w \in MIT_{P_1'}$): In $P_1'\|T$ erhält man $(p_{01}', p_0) \stackrel{x_1 \ldots x_n}{=\!=\!\Rightarrow}_{P_1'\|T} (p', p_n)$ mit $p' \stackrel{x_{n+1}}{\not\longrightarrow}_{P_1'}$ und $p_n \stackrel{x_{n+1}}{\longrightarrow}_T$. Deshalb gilt $(p', p_n) \in E_{P_1'\|T}$. Da alle Aktionen aus w bis auf x_{n+1} synchronisiert werden und $I \cap I_T = \emptyset$, gilt $x_1, \ldots, x_n \in O_{P_1'\|T}$. Es gilt also $\varepsilon \in PrET(P_1'\|T)$.

- Fall 2b) ($w \in PrET_{P_1'}$): In der Parallelkomposition $P_1'\|T$ erhält man die Transitionsfolge $(p_{01}, p_0) \stackrel{w}{\Rightarrow}_{P_1'\|T} (p'', p_{n+1}) \stackrel{u}{\Rightarrow}_{P_1'\|T} (p', p_{n+1})$ für $u \in O^*$ und $p' \in E_{P_1'}$. Daraus folgt $(p', p_{n+1}) \in E_{P_1'\|T}$ und somit $wu \in StET(P_1'\|T)$. Da alle Aktionen in w synchronisiert werden und $I \cap I_T = \emptyset$, gilt $x_1, \ldots, x_n, x_{n+1} \in O_{P_1'\|T}$ und, da $u \in O^*$, folgt $u \in O^*_{P_1'\|T}$. Somit ergibt sich $\varepsilon \in PrET(P_1'\|T)$.

Da $\varepsilon \in PrET(P_1'\|T)$ gilt, kann durch $P_2\, sat_{\text{as}}\, T \Rightarrow P_1\, sat_{\text{as}}\, T$ geschlossen werden, dass auch $P_2\, sat_{\text{as}}\, T$ nicht gelten kann. Die Relation gilt für P_2 nicht, da es eine as-Implementierung P_2' von P_2 gibt, so dass $P_2'\|T$ nicht fehler-frei ist. Da es nur lokale Aktionen in $P_2'\|T$ gibt, muss der Fehler-Zustand lokal erreichbar sein.
Der Fehler kann geerbt oder neu sein.

- Fall 2i) (neuer Fehler): Da jeder Zustand von T alle Inputs $x \in O = I_T$ zulässt, muss ein lokal erreichbarer Fehler-Zustand der Form sein, dass ein Outputs $a \in O_T$ von T möglich ist, der nicht mit einem passenden Input aus P_2' synchronisiert werden werden kann. Durch die Konstruktion von T sind in p_{n+1} keine Outputs möglich. Ein neuer Fehler muss also die Form (p', p_i) haben mit $i \leq n$, $p' \stackrel{x_{i+1}}{\not\longrightarrow}_{P_2'}$ und $x_{i+1} \in O_T = I$. Durch Projektion erhält man dann $p_{02} \stackrel{x_1 \ldots x_i}{=\!=\!\Rightarrow}_{P_2'} p' \stackrel{x_{i+1}}{\not\longrightarrow}_{P_2'}$ und damit gilt $x_1 \ldots x_{i+1} \in MIT_{P_2'} \subseteq ET_{P_2'}$. Es ist also ein Präfix von w in $ET_{P_2'}$ enthalten und mit Proposition 3.6 auch in ET_2.

- Fall 2ii) (geerbter Fehler): T hat $x_1 \ldots x_i u$ mit $u \in I_T^* = O^*$ ausgeführt und ebenso hat P_2' dieses Wort abgearbeitet. Durch dies hat P_2' einen Zustand in $E_{P_2'}$ erreicht, da von T keine Fehler geerbt werden können. Es gilt dann $\text{prune}(x_1 \ldots x_i u) = \text{prune}(x_1 \ldots x_i) \in PrET_{P_2'} \subseteq ET_{P_2'}$. Da $x_1 \ldots x_i$ ein Präfix von w ist, führt in diesem Fall eine Verlängerung um lokale Aktionen von einem Präfix von w zu einem Fehler-Zustand. Da ET der Menge aller Verlängerungen von gekürzten Fehler-Traces entspricht, ist $x_1 \ldots x_i$ in $ET_{P_2'}$ enthalten und somit ist mit Proposition 3.6 ein Präfix von w in ET_2 enthalten.

Um die andere Inklusion zu beweisen, reich es aufgrund der ersten Inklusion und der Definition von EL aus zu zeigen, dass $L_1 \backslash ET_1 \subseteq EL_1$ gilt.
Es wird dafür ein beliebiges $w \in L_1 \backslash ET_1$ gewählt und gezeigt, dass es in EL_2 enthalten ist. Das w ist wegen der Propositionen 2.1 und 3.6 auch für eine as-Implementierung P_1' von P_1 in $L_{P_1'} \backslash ET_{P_1'}$ enthalten.

- Fall 1 ($w = \varepsilon$): Da ε immer in EL_2 muss hier nichts gezeigt werden.

- Fall 2 ($w = x_1 \dots x_n$ mit $n \geq 1$): Es wird ein Partner T wie folgt konstruiert (T entspricht dabei U aus Abbildung 3.2 bis auf die Benennung der Mengen):
  - $T = \{p_0, p_1, \dots, p_n, p\}$,
  - $p_{0T} = p_0$,
  - $\longrightarrow_T = \{(p_j, x_{j+1}, p_{j+1}) \mid 0 \leq j < n\}$
    $\cup \{(p_j, x, p) \mid x \in I_T \backslash \{x_{j+1}\}, 0 \leq j < n\}$
    $\cup \{(p, x, p) \mid x \in I_T\}$,
  - $E_T = \{p_n\}$.

  Da $p_{01} \stackrel{w}{\Rightarrow}_{P_1'} p'$ gilt, kann man schließen, dass $P_1' \| T$ ein lokal erreichbaren geerbten Fehler hat. Es muss also auch eine as-Implementierung P_2' von P_2 geben, für die $P_2' \| T$ einen lokal erreichbaren Fehler-Zustand hat.
  - Fall 2a) (neuer Fehler aufgrund von $x_i \in O_T$ und $p_{02} \stackrel{x_1 \dots x_{i-1}}{\Longrightarrow}_{P_2'} p'' \stackrel{x_i}{\not\longrightarrow}_{P_2'}$): Es gilt $x_1 \dots x_i \in MIT_{P_2'}$ und somit $w \in EL_{P_2'}$. Anzumerken ist, dass es nur auf diesem Weg Outputs von T möglich sind, deshalb gibt es keine anderen Outputs von T, die zu einem neuen Fehler führen können. Es gilt $w \in EL_2$ wegen Proposition 3.6.
  - Fall 2b) (neuer Fehler aufgrund von $a \in O = I_T$): Der einzige Zustand, in dem T nicht alle Inputs erlaubt sind, ist p_n, der bereits ein Fehler-Zustand ist. Da dieser Zustand in $P_2' \| T$ erreichbar ist, besitzt der komponierte MEIO einen geerbten Fehler und es gilt $w \in L_2 \subseteq EL_2$, wegen dem folgenden Fall 2c).
  - Fall 2c) (geerbter Fehler von T): Da p_n der einzige Fehler-Zustand in T ist und alle Aktionen synchronisiert sind, ist dies nur möglich, wenn $p_{02} \stackrel{x_1 \dots x_n}{\Longrightarrow}_{P_2'}$ gilt. In diesem Fall gilt $w \in L_{P_2'} \subseteq EL_{P_2'}$. Daraus folgt mit Proposition 3.6 $w \in EL_2$.
  - Fall 2d) (geerbter Fehler von P_2'): Es gilt dann $p_{02} \stackrel{x_1 \dots x_n}{\Longrightarrow}_{P_2'} p' \in E_{P_2'}$ für $i \geq 0$ und $u \in O^*$. Somit ist $x_1 \dots x_i u \in StET_{P_2'}$ und damit $\text{prune}(x_1 \dots x_i u) = \text{prune}(x_1 \dots x_i) \in PrET_{P_2'} \subseteq EL_{P_2'}$. Somit gilt mit Hilfe von Proposition 3.6 $w \in EL_{P_2'} \subseteq EL_2$.

□

Satz 3.14. *Falls $P_1 \sqsubseteq_E P_2$ gilt folgt draus auch, dass P_1 P_2 verfeinert.*

Beweis. Nach Definition gilt, genau dann wenn $\varepsilon \in ET(P)$, ist ein Fehler-Zustand lokal erreichbar in P. $P_1 \sqsubseteq_E P_2$ impliziert, dass $\varepsilon \in ET_2$ gilt, wenn $\varepsilon \in ET_1$. Somit ist ein Fehler-Zustand in P_1 nur dann lokal erreichbar, wenn dieser auch in P_2 lokal erreichbar ist. Aufgrund von Proposition 3.6 gibt es dann auch entsprechende as-Implementierungen P_1' und P_2' für P_1 bzw. P_2, die ebenfalls Fehler-Zustände lokal erreichen können,

wenn dies in P_1 bzw. P_2 möglich ist. Für einen entsprechenden Test T gilt dann auch, dass $P'_j \| T$ für $j \in \{1, 2\}$ einen lokal erreichbaren Fehler hat, wenn P'_j einen solchen hat. Falls also P_1 einen lokal erreichbaren Fehler-Zustand enthält, dann zeigt sich dies Verhalten auch in einer as-Implementierungen und auch in der Parallelkomposition der as-Implementierung mit einem Test T. Aus einem lokal erreichbaren Fehler in P_1 folgt auch ein lokal erreichbarer Fehler-Zustand in P_2 und somit auch ein lokal erreichbarer Fehler-Zustand in der Parallelkomposition einer as-Implementierung von P_2 mit einem Test T. Aus $P_1 \sqsubseteq_E P_2$ folgt also die Implikation $\neg P_1 \, sat_{\text{as}} \, T \Rightarrow \neg P_2 \, sat_{\text{as}} \, T$ für alle Test T. Wenn man die Negationen entfernt, ergibt sich für alle Tests T: $P_2 \, sat_{\text{as}} \, T \Rightarrow P_1 \, sat_{\text{as}} \, T$. Dies entspricht der Definition von P_1 verfeinert P_2. □

Auch die in diesem Abschnitt gezeigten Folgerungen schließen sich zu einem Ring. Dies ist in Abbildung 3.4 dargestellt.

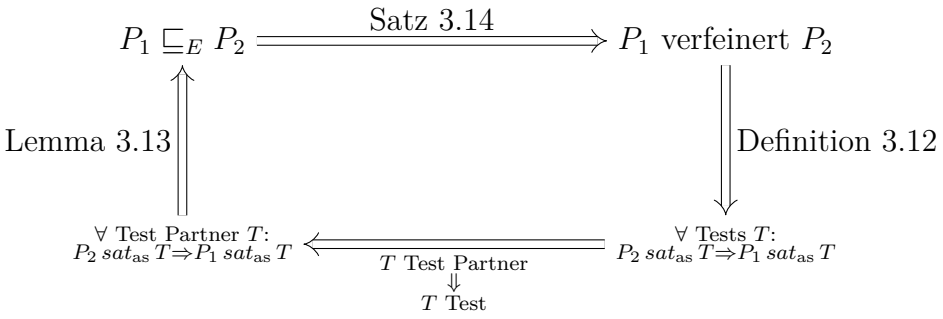

Abbildung 3.4: Folgerungskette der Testing-Verfeinerung und Fehler-Relation

4 Verfeinerungen für Kommunikationsfehler- und Ruhe-Freiheit

In diesem Kapitel wird die Menge der betrachteten Zustandsmengen von den Kommunikations-basierten Fehlern im letzten Kapitel erweitert um Ruhe-Zustände.
Zustände, die keine must-Outputs ohne einen Input ausführen können, werden als in einer Art Verklemmung angesehen, da sie ohne Zutun von Außen den Zustand nicht mehr verlassen können, falls ein möglicherweise vorhandener may-Output nicht implementiert wird. So ein Zustand hat also keine must-Transitions-Möglichkeiten für einen Output. Falls dieser Zustand die Möglichkeit für eine interne Aktion via einer must-Transition hat, darf durch die τs niemals ein Zustand erreicht werden, von dem aus ein Output in Implementierungen sicher gestellt wird. Ein Zustand, der keine Outputs und τs via must-Transitionen ausführen kann, ist also ein Deadlock-Zustand, in denen das System nichts mehr tun können muss ohne einen Input. Wenn man eine Erweiterung um τs zu Zuständen ohne must-Outputs zulässt, hat man zusätzlich noch Verklemmungen der Art Livelock, da diese Zustände möglicherweise beliebig viele interne Aktionen ausführen können, jedoch nie aus eigener Kraft einen wirklichen Fortschritt in Form eines Outputs bewirken können müssen. Die Menge der Zustände, die sich in einer Verklemmung befinden, würde also durch $\left\{p \in P \mid \forall a \in O : p \overset{a}{\Rightarrow\!\!\!\!\!/}_P\right\}$ beschrieben werden. Somit wären dies alle Zustände, die keine Möglichkeit haben ohne einen Input von Außen oder eine implementierte may-Output-Transition je wieder einen Output machen zu können. Falls man diese Definition verwenden würde, müsste man immer alle Zustände betrachten, die durch τs erreichbar sind. Dies würde einige Betrachtungen deutlich aufwendiger machen und soll deshalb hier nicht behandelt werden. Die Definition für die betrachteten Verklemmungen, hier Ruhe genannt, beschränkt sich auf Zustände, die keine Outputs und τs via must-Transitionen ausführen können.

Definition 4.1 (*Ruhe*). *Ein* Ruhe-Zustand *ist ein Zustand in einem MEIO P, der keine Outputs und kein τ zulässt via must-Transitionen.*
Somit ist die Menge der Ruhe-Zustände in einem MEIO P wie folgt formal definiert: $Qui(P) := \left\{p \in P \mid \forall \alpha \in (O \cup \{\tau\}) : p \overset{\alpha}{\not\rightarrow}_P\right\}$.

Für die Erreichbarkeit wird wie im letzten Kapitel ein optimistischer Anzahl der lokalen Erreichbarkeit für die Fehler-Zustände verwendet. Ruhe ist kein unabwendbare „Fehler-Art“, sondern kann durch einen Input repariert werden oder im Fall von vorhandenen

may-Output-Transitionen oder may-τ-Transitionen, durch eine Implementierung dieses Outputs oder des τs. Daraus ergibt sich, dass Ruhe im Vergleich zu den Fehlern aus dem letzten Kapitel als weniger „schlimmer Fehler“ anzusehen ist. Somit ist ein Ruhe-Zustand ebenso wie ein Fehler-Zustand erreichbar, sobald er durch Outputs und τs erreicht werden kann, jedoch ist nicht jede beliebige Fortsetzung eines Traces, das durch lokale Aktionen zu einem Ruhe-Zustand führt ein Ruhe-Trace.

Definition 4.2 (*fehler- und ruhe-freie Kommunikation*). *Zwei MEIOs P_1 und P_2* kommunizieren fehler- und ruhe-frei*, wenn keine as-Implementierung ihrer Parallelkomposition P_{12} einen Fehler- oder Ruhe-Zustand lokal erreichen kann.*

TODO: durch entsprechende Testing-Aussagen ersetzten

Definition 4.3 (*Ruhe-Verfeinerungs-Basisrelation*). *Für MEIOs P_1 und P_2 mit der gleichen Signatur wird $P_1 \sqsubseteq^{\mathrm{B}}_{Qui} P_2$ geschrieben, wenn ein Fehler- oder Ruhe-Zustand in einer as-Implementierung von P_1 nur dann lokal erreichbar ist, wenn es auch eine as-Implementierung von P_2 gibt, in der ein Fehler oder ruhiger Zustand lokal erreichbar ist. Diese* Basisrelation *stellt eine* Verfeinerungs-Relation *bezüglich* Fehler- *und* Ruhe-Freiheit *dar.*
$\sqsubseteq^{\mathrm{C}}_{Qui}$ bezeichnet die vollständig abstrakte Präkongruenz *von $\sqsubseteq^{\mathrm{B}}_{Qui}$ bezüglich $\cdot \| \cdot$.*

Um eine genauere Auseinandersetzung mit den Präkongruenzen zu ermöglichen, benötigt man wie im letzten Kapitel die Definition von Traces auf der Struktur. Dadurch erhält man die Möglichkeit die gröbste Präkongruenz charakterisieren zu können. Wie bereits oben erwähnt, ist Ruhe ein reparierbares „Fehlverhalten“ im Gegensatz zu Fehlern. Es genügt deshalb für Ruhe die strikten Traces ohne Kürzung zu betrachten.

Definition 4.4 (*Ruhe-Traces*). *Sei P ein MEIO und definiere:*

- strikte Ruhe-Traces*: $StQT(P) := \left\{ w \in \Sigma^* \mid p_0 \stackrel{w}{\Rightarrow}_P p \in Qui(P) \right\}$.*

Proposition 4.5 (*Ruhe-Traces und Implementierungen*). *Für ein MEIO P gilt für die strikte Ruhe-Traces: $StQT(P) \subseteq \left\{ w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \in Qui(P') \right\} = \bigcup_{P' \in \text{as-impl}(P)} StQT(P')$.*

Beweis. Analog zu den Propositionen 2.1 und 3.4 ist die Inklusion am Besten mit einer as-Implementierung zu zeigen und der entsprechenden as-Verfeinerungs-Relation $\mathcal{R}$. Falls ε zu einem Ruhe-Zustand führt in P, muss man zwei as-Implementierungen betrachten, ansonsten genügt es eine für alle w aus $StQT(P)$ anzugeben.
Die as-Implementierung P' für den Fall $\varepsilon \in StQT(P)$ implementiert alle must-Transitionen, keine may-Transitionen und keine Fehler-Zustände von P und hat die Identitäts-Relation als starke as-Verfeinerungs-Relation $\mathcal{R}$. Für alle $a \in \Sigma$ gilt somit, wenn in P für einen Zustand p $p \stackrel{a}{\not\rightarrow}$ galt auch in P' $p' \stackrel{a}{\not\rightarrow}$ für den Zustand, der mit p in Relation steht. Der Startzustand von P' ist mit ε erreichbar und auch ruhig. Es gilt also $\varepsilon \in StQT(P')$.

Der 2. Punkt der Definition 1.3 ist für die Identitäts-Relation als as-Verfeinerungs-Relation $\mathcal{R}$ erfüllt, da alle must-Transitionen aus P entsprechend in P' umgesetzt wurden. Alle must-Transitionen in P müssen zugrundeliegende may-Transitionen haben, somit gilt auch 3. von 1.3. Der 1. Punkt der Definition ist auch erfüllt, da $E_{P'} = \emptyset$ gilt, wenn keine Fehler-Zustände implementiert werden.
Für alle $w \neq \varepsilon$ mit $w \in StQT(P)$ kann P' als die folgende as-Implementierung gewählt werden:

- $P' = \{q \mid p \in P\} \cup \{q' \mid p \in P\}$,
- $p'_0 = q_0$,
- $I_{P'} = I_P$ und $O_{P'} = O_P$,
- $\longrightarrow_{P'} = \dashrightarrow_{P'} = \Big\{(q_0, \alpha, q_j) \mid p_0 \overset{\alpha}{\dashrightarrow} p_j\Big\}$
  $\cup \Big\{(q_0, \alpha, q'_j) \mid p_0 \overset{\alpha}{\dashrightarrow} p_j\Big\}$
  $\cup \Big\{(q_j, \alpha, q_k) \mid p_j \overset{\alpha}{\longrightarrow} p_k\Big\}$
  $\cup \Big\{(q'_j, \alpha, q'_k) \mid p_j \overset{\alpha}{\longrightarrow} p_k\Big\}$
  $\cup \Big\{(q'_j, \alpha, q_k) \mid j \neq 0, p_j \overset{\alpha}{\dashrightarrow} p_k, p_j \overset{\alpha}{\not\longrightarrow} p_k\Big\}$
  $\cup \Big\{(q'_j, \alpha, q'_k) \mid j \neq 0, p_j \overset{\alpha}{\dashrightarrow} p_k, p_j \overset{\alpha}{\not\longrightarrow} p_k\Big\}$,
- $E_{P'} = \emptyset$.

Als as-Verfeinerungs-Relation zwischen P und P' wird die Relation $\mathcal{R} = \{(q_j, p_j) \mid p_j \in P\} \cup \{(q'_j, p_j) \mid p_j \in P\}$ verwendet. Es werden in P' für die ungestrichenen Zustände q nur die must-Transitionen und für die gestrichenen Zustände q' werden die must- und may-Transitionen implementiert. Die must-Transitionen werden nur zu den Zuständen der „gleichen Sorte“ umgesetzt, wohingegen die may-Transitionen, zu denen es keine entsprechende must-Transition in P gibt, von den Zuständen q' zu dem entsprechenden ungestrichenen und gestrichenen Zustand implementiert wird. Da die Menge der Fehler-Zustände leer ist, gilt 1.3 1. für $\mathcal{R}$. Die must-Transitionen werden für die ungestrichenen und gestrichenen Zustände umgesetzt, dies erfüllt zusammen mit $\mathcal{R}$ die Definition 1.3 2. Ebenso wird der dritte Punkt dieser Definition erfüllt, da sowohl die gestrichenen wie auch die ungestrichenen Zustände mit den entsprechenden Zuständen aus P in der Relation $\mathcal{R}$ stehen. $\mathcal{R}$ ist also eine starke alternierende Simulations-Relation auf P' und P. Falls ein Zustand p_j in P ruhig war, ist es auch der entsprechenden Zustand q_j in P', da für q_j alle ausgehenden must-Transitionen von p_j implementiert wurden, aber keine einzige may-Transition, die keine der must-Transitionen entspricht. Wenn also für p_j keine Outputs und kein τ möglich waren via must-Transitionen, dann ist es dies auch für q_j nicht. q_j ist in P' mit den selben Traces erreichbar wie p_j in P, da jeder ungestrichene und gestrichene Zustand in P' die selben eingehenden Transitionen hat wie der entsprechende Zustand in P. Falls der Trace zu p_j may-Transitionen ohne entsprechende must-Transitionen enthält, kann der entsprechende Trace in P' ausgeführt werden, in dem von q_0 aus der Trace über die gestrichenen Zustände genommen wird bis zur letzten

Transition, die zu einem ungestrichenen Zuständen führt in dem auszuführenden Wort. Ab da hat der Trace in P nur must-Transitionen genommen und kann somit in den ungestrichenen Zuständen in P' nachgefolgt werden. Falls der Trace in P insgesamt nur aus must-Transitionen bestanden hat, ist direkt von q_0 aus der Weg über ungestrichene Zustände zu q_j möglich. Es gilt also $StQT(P)\backslash\{\varepsilon\} = StQT(P')\backslash\{\varepsilon\}$. □

Für ET und EL gelten die Definitionen aus dem letzten Kapitel. Es wir nur für Ruhe eine neue Semantik definiert.

Definition 4.6 (*Ruhe-Semantik*). *Sei P ein MEIO.*

- *Die Menge der* fehler-gefluteten Ruhe-Traces *von P ist $QET(P) := StQT(P) \cup ET(P)$.*

Für zwei MEIOs P_1, P_2 mit der gleichen Signatur wird $P_1 \sqsubseteq_{Qui} P_2$ geschrieben, wenn $P_1 \sqsubseteq_E P_2$ und $QET_1 \subseteq QET_2$ gilt.

Proposition 4.7 (*Ruhe-Semanik und Implementierungen*). *Für die Menge der fehler-gefluteten Ruhe-Traces von P gilt $QET(P) = \bigcup_{P' \in \text{as-impl}(P)} QET(P')$.*

Beweis.
$\subseteq$:

$$\begin{aligned} QET(P) &\overset{4.6}{=} StQT(P) \cup ET(P) \\ &\overset{4.5}{\subseteq} \left(\bigcup_{P' \in \text{as-impl}(P)} StET(P') \right) \cup ET(P) \\ &\overset{3.6}{=} \left(\bigcup_{P' \in \text{as-impl}(P)} StET(P') \right) \cup \left(\bigcup_{P' \in \text{as-impl}(P)} ET(P') \right) \\ &= \bigcup_{P' \in \text{as-impl}(P)} StQT(P') \cup ET(P') \\ &\overset{4.6}{=} \bigcup_{P' \in \text{as-impl}(P)} QET(P'). \end{aligned}$$

$\supseteq$:
Es wird hier für ein $w \in QET(P')$ einer beliebigen as-Implementierung P' von P gezeigt, dass das Wort w auch in $QET(P)$ enthalten ist. Es kann danach unterschieden werden, ob w aus $StQT(P')\backslash ET(P')$ stammt oder aus $ET(P')$. Falls $w \in ET(P')$ gilt, folgt mit Proposition 3.6 bereits, dass $w \in ET(P) \subseteq QET(P)$ gilt. Somit wird für den Rest des Beweises davon ausgegangen, dass $w \in StQT(P')\backslash ET(P')$ ist. w führt in P' also nur zu einem ruhigen Zustand und hat nichts mit Fehler-Zuständen in P' zu tun. Da es eine as-Verfeinerungs-Relation $\mathcal{R}$ geben muss, die beweist, dass P' P as-verfeinert,

muss es ein Präfix von w geben, dass auch in P ausführbar ist. Falls w nicht vollständig ausführbar ist in P, muss auf dem Weg, auf dem das Präfix von w ausgeführt wird ein Zustand liegen, der ein Fehler-Zustand ist. Es gilt dann $w \in ET(P) \subseteq QET(P)$. Falls jedoch w in P ausführbar ist ohne einen Fehler-Zustand zu erreichen, musst der dadurch erreichte Zustand p mit dem Zustand p', der in P' durch w erreicht wird, in der starken as-Verfeinerungs-Relation $\mathcal{R}$ stehen. p' ist ruhig, nach Voraussetzung, dass $w \in StQT(P')$ enthalten ist. Es gilt also für alle $\omega \in O \cup \{\tau\}$ $p' \overset{\omega}{\not\longrightarrow}$. Da $(p', p) \in \mathcal{R}$ in Relation stehen und beide Zustände keine Fehler-Zustände sind, muss auch $p \overset{\omega}{\not\longrightarrow}$ für alle $\omega \in O \cup \{\tau\}$ gelten, da sonst 1.3 2. verletzt würde. Es gilt also in diesem Fall $w \in StQT(P) \subseteq QET(P)$. □

Lemma 4.8 (*Ruhe-Zustände unter Parallelkomposition*).

1. *Ein Zustand (p_1, p_2) aus der Parallelkomposition P_{12} ist ruhig, wenn es auch die Zustände p_1 und p_2 in P_1 bzw. P_2 sind.*
2. *Wenn der Zustand (p_1, p_2) ruhig ist und nicht in E_{12} enthalten ist, dann sind auch die auf die Teilsysteme projizierten Zustände p_1 und p_2 ruhig.*

Beweis.

1. Da $p_1 \in Qui_1$ und $p_2 \in Qui_2$ gilt, haben diese beiden Zustände jeweils höchstens die Möglichkeit für Input-Transitionen oder Output- und τ-may-Transitionen, jedoch keine Möglichkeit für Outputs oder τs als must-Transitionen.
   Angenommen der Zustand, der durch die Parallelkomposition aus den Zuständen p_1 und p_2 entsteht, ist nicht ruhig, d.h. er hat eine ausgehende must-Transition für einen Output oder ein τ.
   - Fall 1 $\left((p_1, p_2) \overset{\tau}{\longrightarrow}_{12}\right)$: Ein τ ist eine interne Aktion und kann in der Parallelkomposition nicht durch das Verbergen von Aktionen bei der Synchronisation entstehen. Ein τ in der Parallelkomposition ist also auch nur möglich, wenn dies bereits als must-Transition in einer Komponente möglich war für einen der Zustände, aus denen (p_1, p_2) zusammensetzt ist. Jedoch verbietet die Voraussetzung, dass p_1 oder p_2 eine ausgehende τ must-Transition haben, deshalb kann auch (p_1, p_2) keine solche Transition besitzen.
   - Fall 2 $\left((p_1, p_2) \overset{a}{\longrightarrow}_{12} \text{ mit } a \in O_{12} \backslash \text{Synch}(P_1, P_2)\right)$: Da es sich bei a um einen Output handelt, der nicht in $\text{Synch}(P_1, P_2)$ enthalten ist, kann dieser nicht aus der Synchronisation von zwei Aktionen entstanden sein, sondern muss bereits für P_1 oder P_2 als must-Transition ausführbar gewesen sein. Es gilt also oBdA $p_1 \overset{a}{\longrightarrow}_1$ mit $a \in O_1$. Dies ist jedoch aufgrund der Voraussetzung nicht möglich. Somit kann die Parallelkomposition diese Transition für (p_1, p_2) ebenfalls nicht als must-Transition enthalten.
   - Fall 3 $\left((p_1, p_2) \overset{a}{\longrightarrow}_{12} \text{ mit } a \in O_{12} \cap \text{Synch}(P_1, P_2)\right)$: Der Output a ist in diesem Fall durch Synchronisation von einem Output mit einem Input entstanden. OBdA gilt $a \in O_1 \cap I_2$. Für die einzelnen Systeme muss also gelten, dass

$p_1 \xrightarrow{a}_1$ und $p_2 \xrightarrow{a}_2$. Die Transition für das System P_1 ist jedoch in der Voraussetzung ausgeschlossen worden. Somit ist es nicht möglich, dass P_{12} diese in diesem Fall angenommene must-Transition für den Zustand (p_1, p_2) ausführen kann.

Da alle diese Fälle zu einem Widerspruch mit der Voraussetzung führen folgt, dass bereits die Annahme, dass der Zustand (p_1, p_2) nicht ruhig ist, falsch war. Es gilt also, dass aus $p_j \in Qui_j$ für $j \in \{1, 2\}$ $(p_1, p_2) \in Qui_{12}$ folgt.

2. Es gilt $(p_1, p_2) \in Qui_{12} \backslash E_{12}$, somit hat dieser Zustand allenfalls die Möglichkeit für must-Transitionen, die mit Inputs beschriftet sind.
Angenommen $p_1 \notin Qui_1$, dann ist für p_1 entweder eine τ-must-Transition oder eine Output-must-Transition möglich.

   - Fall 1 $\left(p_1 \xrightarrow{\tau}_1\right)$: Da die Transition für P_1 möglich ist, hat auch P_{12} die Möglichkeit für eine τ-must-Transition. Dies ist jedoch durch die Voraussetzung verboten und somit kann dieser Fall nicht eintreten.
   - Fall 2 $\left(p_1 \xrightarrow{a}_1 \text{ mit } a \in O_1 \backslash \text{Synch}(P_1, P_2)\right)$: Da es sich bei a um einen must-Output handelt, der nicht zu synchronisieren ist, wird dieser einfach in die Parallelkomposition übernommen. Es müsste also $(p_1, p_2) \xrightarrow{a}_{12}$ mit $a \in O_{12}$ gelten, was jedoch verboten ist. Somit kann die Transition für P_1 in diesem Fall nicht möglich sein.
   - Fall 3 $\left(p_1 \xrightarrow{a}_1 \text{ mit } a \in O_1 \cap \text{Synch}(P_1, P_2) \text{ und } p_2 \xrightarrow{a}_2\right)$: In diesem Fall ist die Synchronisation des Outputs a von P_1 mit dem Input a von P_2 möglich, so dass in der Parallelkomposition der Output a als must-Transition für (p_1, p_2) entsteht. Diese must-Transition ist jedoch für P_{12} nach Voraussetzung nicht erlaubt. Es folgt also auch, dass dieser Fall nicht eintreten kann.
   - Fall 4 $\left(p_1 \xrightarrow{a}_1 \text{ mit } a \in O_1 \cap \text{Synch}(P_1, P_2) \text{ und } p_2 \not\xrightarrow{a}_2\right)$: Da P_2 die a Transition nicht als must-Transition enthält, handelt es sich hier um einen neuen Fehler. Das a kann für P_2 kein Output sein, da sonst P_1 und P_2 nicht komponierbar wäre. Der neue Fehler kann dadurch entstehen, dass die Synchronisation des Outputs a von P_1 mit dem Input a von P_2 an dieser Stelle nicht möglich ist, oder da der Input a für p_2 nur als may-Transition vorliegt und somit die Gefahr besteht, dass dieser in einer Implementierung nicht vorhanden ist. Im zweiten Fall synchronisieren die beiden Transitionen zu einer a Output-may-Transition, die in P_{12} zulässig wäre. Jedoch wird der Zustand (p_1, p_2) in beiden Fällen in die Menge E_{12} der Parallelkomposition eingefügt (Definition 1.2). Dies wurde in der Voraussetzung für den Zustand ausgeschlossen und dieser Fall ist somit nicht möglich.

   Alle aufgeführten Fälle führen zu einem Widerspruch mit der Voraussetzung, somit folgt, dass die Annahme bereits falsch war und $p_1 \in Qui_1$ gelten muss. Analog kann für p_2 argumentiert werden, so dass dann auch $p_2 \in Qui_2$ folgt.

□

In dem folgenden Satz sind die Punkte 1. und 3. nur zur Vollständigkeit aufgeführt. Sie entsprechen Punkt 1. und 2. aus Satz 3.7.

Satz 4.9 (*Kommunikationsfehler- und Ruhe-Semantik für Parallelkompositionen*). *Für zwei komponierbare MEIOs P_1, P_2 und ihre Komposition P_{12} gilt:*

1. $ET_{12} = \text{cont}(\text{prune}((ET_1 \| EL_2) \cup (EL_1 \| ET_2)))$,
2. $QET_{12} = (QET_1 \| QET_2) \cup ET_{12}$,
3. $EL_{12} = (EL_1 \| EL_2) \cup ET_{12}$.

Beweis. Es wird nur der 2. Punkt beweisen.
„$\subseteq$“:
Hier muss unterschieden werden, ob ein $w \in StQT_{12} \backslash ET_{12}$ oder ein $w \in ET_{12}$ betrachtet wird. Im zweiten Fall ist das w offensichtlich in der rechten Seite enthalten. Somit wird ab jetzt ein $w \in StQT_{12} \backslash ET_{12}$ betrachtet und es wird versucht dessen Zugehörigkeit zur rechten Menge zu zeigen. Aufgrund von Definition 4.4 weiß man, dass $(p_{01}, p_{02}) \stackrel{w}{\Rightarrow}_{12} (p_1, p_2)$ gilt mit $(p_1, p_2) \in Qui_{12} \backslash E_{12}$. Durch Projektion erhält man $p_{01} \stackrel{w_1}{\Rightarrow}_1 p_1$ und $p_{02} \stackrel{w_2}{\Rightarrow}_2 p_2$ mit $w \in w_1 \| w_2$. Aus $(p_1, p_2) \in Qui_{12} \backslash E_{12}$ kann mit dem zweiten Punkt von Lemma 4.8 gefolgert werden, dass bereits $q_1 \in Qui_1$ und $q_2 \in Qui_2$ gilt. Somit gilt $w_1 \in StQT_1 \subseteq QET_1$ und $w_2 \in StQT_2 \subseteq QET_2$. Daraus folgt dann $w \in QET_1 \| QET_2$ und somit ist w in der rechten Seite der Gleichung enthalten.

„$\supseteq$“:
Es muss wieder danach unterscheiden werden aus welcher Menge das betrachtete Element stammt. Falls $w \in ET_{12}$ gilt, so kann die Zugehörigkeit zur linken Seite direkt gefolgert werden. Somit wird für den weiteren Beweis dieser Inklusionsrichtung ein Element $w \in QET_1 \| QET_2$ betrachtet und gezeigt, dass es in der linken Menge enthalten ist. Da $QET_i = StQT_i \cup ET_i$ gilt, existieren für w_1 und w_2 mit $w \in w_1 \| w_2$ unterschiedliche Möglichkeiten:

- Fall 1 ($w_1 \in ET_1 \vee w_2 \in ET_2$): OBdA gilt $w_1 \in ET_1$. Nun kann $w_2 \in StQT_2 \subseteq L_2$ oder $w_2 \in ET_2$ gelten und somit ist auf jeden Fall w_2 in EL_2 enthalten. Daraus kann dann mit dem ersten Punkt von Satz 3.7 gefolgert werden, dass $w \in ET_{12}$ gilt und damit ist w in der linken Seite der Gleichung enthalten.
- Fall 2 ($w_1 \in StQT_1 \backslash ET_1 \wedge w_2 \in StQT_2 \backslash ET_2$): Es gilt in diesem Fall $p_{01} \stackrel{w_1}{\Rightarrow}_1 p_1 \in Qui_1$ und $p_{02} \stackrel{w_2}{\Rightarrow}_2 p_2 \in Qui_2$. Da p_1 und p_2 in der jeweiligen Ruhe-Menge enthalten sind, ist auch der Zustand, der aus ihnen zusammengesetzt ist, in der Parallelkomposition ruhig, wie bereits im ersten Punkt von Lemma 4.8 gezeigt. Es gilt also für die Komposition $(p_{01}, p_{02}) \stackrel{w}{\Rightarrow}_{12} (p_1, p_2) \in Qui_{12}$ und dadurch ist w in der linken Seite der Gleichung enthalten, da $w \in StQT_{12} \subseteq QET_{12}$ gilt.

□

Korollar 4.10 (*Ruhe-Präkongruenz*). *Die Relation $\sqsubseteq_{Qui}$ ist eine Präkongruenz bezüglich $\cdot \| \cdot$.*

Beweis. Es muss gezeigt werden: Wenn $P_1 \sqsubseteq_{Qui} P_2$ gilt, so auch $P_{31} \sqsubseteq_{Qui} P_{32}$ für jedes komponierbare System P_3. D.h. es ist zu zeigen, dass aus $P_1 \sqsubseteq_E P_2$ und $QET_1 \subseteq QET_2$ sowohl $P_{31} \sqsubseteq_E P_{32}$ als auch $QET_{31} \subseteq QET_{32}$ folgt. Dies ergibt sich, wie im Beweis zu Korollar 3.8, aus der Monotonie von $\cdot \| \cdot$ auf Sprachen wie folgt:

- $P_{31} \overset{\substack{\text{Korollar 3.8} \\ \text{und} \\ P_1 \sqsubseteq_E P_2}}{\sqsubseteq_E} P_{32}$,
- $QET_{31} \overset{4.9\ 2.}{=} (QET_3 \| QET_1) \cup ET_{31}$
  $\overset{\substack{ET_{31} \subseteq ET_{32} \\ \text{und} \\ QET_1 \subseteq QET_2}}{\subseteq} (QET_3 \| QET_2) \cup ET_{32}$
  $\overset{4.9\ 2.}{=} QET_{32}$. □

TODO: Verfeinerung auf Basis des Testing-Ansatzes

Literaturverzeichnis

[BFLV16] Ferenc Bujtor, Sascha Fendrich, Gerald Lüttgen, und Walter Vogler, *Nondeterministic Modal Interfaces*, Theor. Comput. Sci. **642** (2016), 24–53.

[BV15a] Ferenc Bujtor und Walter Vogler, *Error-pruning in interface automata*, Theor. Comput. Sci. **597** (2015), 18–39.

[BV15b] ———, *Failure Semantics for Modal Transition Systems*, ACM Trans. Embedded Comput. Syst. **14** (2015), no. 4, 67:1–67:30.

[Sch16] Ayleen Schinko, *Kommunikationsfehler, Verklemmung und Divergenz bei Interface-Automaten*, Bachelorarbeit, Universität Augsburg, 2016.