1 Definitionen

Stand: 20. Juli 2017

Kombination aus [BV15] und [Sch16] mit Einflüssen von [BFLV16]:

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 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.
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 wir 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 zu $p \overset{w}{\dashrightarrow} p'$ ($p \overset{w}{\dashrightarrow}$) 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 wir $p \overset{w}{\Longrightarrow} p'$ 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 $\nexists p' : p \overset{a}{\longrightarrow} p'$ und $\nexists p' : p \overset{a}{\dashrightarrow} p'$ stehen soll. $p \overset{a}{\longrightarrow\!\!\Longrightarrow}$ wird geschrieben, wenn sowohl ein p' wie auch ein p'' existiert, so dass $p \overset{a}{\longrightarrow} p' \overset{\varepsilon}{\Longrightarrow} p''$ gilt. Diese Transition wird auch als *schwache-nachlaufende must-Transition* bezeichnet. Entsprechen steht $\overset{a}{\dashrightarrow\!\!\Longrightarrow}$ für die *schwache-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

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

Dabei bezeichnet $\mathrm{Synch}(P_1, P_2) = (I_1 \cap O_2) \cup (O_1 \cap I_2) \cup (I_1 \cap I_2)$ *die Menge der zu* synchronisierenden Aktionen*. Die synchronisierten Aktionen werden als Output bzw. Input der Komposition beibehalten.*

P_1 wird Partner *von P_2 genannt, wenn die Parallelkomposition von P_1 und P_2 geschlossenen ist, d.h. sie duale Signaturen* $\mathrm{Sig}(P_1) = (I, O)$ *und* $\mathrm{Sig}(P_2) = (O, I)$ *haben.*

Ein neuer Kommunikationsfehler entsteht, wenn eines der MEIOs die Möglichkeit für einen Output hat (may-Transition) und das andere MEIO den passenden Input nicht erzwingt (keine must-Transition vorhanden). Es muss also in möglichen Implementierungen nicht wirklich zu diesem Kommunikationsfehler kommen, da die Output-Transition nicht zwingendermaßen implementiert werden muss und die Input-Transition durch eine may-Transition trotzdem erlaubt sein kann.
Wie bereits in [Sch16] kann es durch die Synchronisation von Inputs zu keinen neuen Kommunikationsfehler kommen, da dies in beiden Transitionssystemen keine lokal kontrollierte Aktion ist. 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 einen neuen Kommunikationsfehler kommen.

Definition 1.3 (*(starke) Simulation*). *Eine* (starke) alternierende Simulation *ist eine Relation $R \subseteq P \times Q$ auf zwei MEIOs P und Q, falls für alle $(p, q) \in R$ gilt:*

1. $q \xrightarrow{\alpha}_Q q'$ *impliziert* $p \xrightarrow{\alpha}_P p'$ *für ein* p' *mit* $p' R q'$,
2. $p \dashrightarrow{}^{\alpha}_P p'$ *impliziert* $q \dashrightarrow{}^{\alpha}_Q q'$ *für ein* q' *mit* $p' R q'$,
3. $p \in E_P \Rightarrow q \in E_Q$.

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, und P* as-verfeinert *Q* (stark) *oder P ist eine* (starke) as-Verfeinerung *von Q.*
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.*

Definition 1.4 (*schwache Simulation*). *Eine* schwache alternierende Simulation *ist eine Relation $R \subseteq P \times Q$ auf zwei MEIOs P und Q, falls für alle $(p, q) \in R$ gilt:*

1. $q \xrightarrow{i}_Q q'$ *impliziert* $p \xrightarrow{i}_P \overset{\varepsilon}{\Longrightarrow}_P p'$ *für ein* p' *mit* $p' R q'$,
2. $q \xrightarrow{\omega}_Q q'$ *impliziert* $p \overset{\hat{\omega}}{\Longrightarrow}_P p'$ *für ein* p' *mit* $p' R q'$,
3. $p \dashrightarrow{}^{i}_P p'$ *impliziert* $q \dashrightarrow{}^{i}_Q \overset{\varepsilon}{\Longrightarrow}_Q q'$ *für ein* q' *mit* $p' R q'$,
4. $p \dashrightarrow{}^{\omega}_P p'$ *impliziert* $q \overset{\hat{\omega}}{\Longrightarrow}_Q q'$ *für ein* q' *mit* $p' R q'$,
5. $p \in E_P \Rightarrow q \in E_Q$.

Wobei $i \in I$ und $\omega \in O \cup \{\tau\}$.
Analog zur starken alternierenden Simulation, wird hier $\sqsubseteq_{\text{w-as}}$ als Relationssymbol verwendet und man kann auch entsprechend schwache as-Verfeinerung *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. Output hingeben können auch verzögert werden, da die Umgebung dies dann als Input aufnimmt und für diese somit nicht lokal kontrolliert ist.

Die Parallelkomposition von Wörtern und Mengen kann aus [Sch16] 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 aus [Sch16] 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* $L(P) := \left\{w \in \Sigma^* \mid p_0 \overset{w}{\Rightarrow}_P\right\}$.

2 allgemeine Folgerungen

Proposition 2.1 (*Sprache und Implementierung*). *Die* (maximale) Sprache *eines MEIOs P ist* $L(P) = \left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}_{P'}\right\}$.

Beweis. Für ein $w \in L(P)$ gilt nach Definition 1.7 und den Definitionen der Transitions-Notation $\exists p_1, p_2, \ldots, p_{n-1}, p' \exists w' \in \Sigma_\tau^* : \hat{w'} = w \wedge w' = \alpha_1\alpha_2 \ldots \alpha_n \wedge p_0 \overset{\alpha_1}{\dashrightarrow}_P p_1 \overset{\alpha_2}{\dashrightarrow}_P \ldots p_{n-1} \overset{\alpha_n}{\dashrightarrow}_P p'$. Für ein w aus $\left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}_{P'}\right\}$ gilt, für ein $P' \in \text{as-impl}(P)$ das analoge nur mit must- anstatt may-Transitionen.
Aufgrund von Definition 1.3 2. kann jedes Element aus as-impl(P) nur die bereits in P vorhandenen may-Transitionen implementieren. Somit gibt es für jedes w, dass in der Sprache einer as-Implementierung von P enthalten ist auch ein entsprechendes $w \in L(P)$ mit einem Trace wie oben.
Da in $\left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \overset{w}{\Longrightarrow}'_P\right\}$ alle Wörter enthalten sind, für dies es eine as-Implementierung gibt, die dieses Wort ausführen kann, werden somit auch alle möglichen as-Implementierungen betrachtet. Jede may-Transition aus P wird von mindestens einem $P' \in \text{as-impl}(P)$ als must-Transition implementiert. Deshalb sind auch alle Wörter, die in $L(P)$ enthalten sind in der Menge der Wörter alle as-Implementierungen von P enthalten.
Da für Implementierungen die must-Transitions-Relations-Menge die gleiche ist, wie die Menge der may-Transitions-Relationen könnte man auch für die as-Implementierungen die Definition 1.7 anwenden um die jeweilige Sprache zu bestimmen. Die Sprache eines MEIO entspricht dann der Vereinigung der Sprachen seiner as-Implementierungen. □

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, kann auf 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 (*as-Implementierungen und Parallelkomposition*).
$P'_1 \in \text{as-impl}(P_1) \wedge P'_2 \in \text{as-impl}(P_2) \Rightarrow (P'_1 \| P'_2) \in \text{as-impl}(P_1 \| P_2)$.

Beweis. Es gelte $j \in \{1, 2\}$. Da $P'_j \in \text{as-impl}(P_j)$ gilt, gibt es nach Definition 1.3 eine as-Verfeinerungs-Relation $\mathcal{R}_j$, die beschreibt, wie P'_j P_j verfeinert. Die Parallelkomposition 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 auch eine neue as-Verfeinerungs-Relation für die Verfeinerung von $P_1 \| P_2$ durch $P'_1 \| P'_2$ erstellen. Die neue 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 Verfeinerungs-Relation nach Definition 1.3 ist, da die Parallelkomposition von zwei Implementierungen auch immer eine Implementierung ist (1.2).

1. Für diesen Punkt der Simulations-Definition 1.3 ist folgendes zu zeigen: $(p_1, p_2) \xrightarrow{\alpha}_{12} (q_1, q_2)$ impliziert $(p'_1, p'_2) \xrightarrow{\alpha}_{P'_1 \| P'_2} (q'_1, q'_2)$ für ein (q'_1, q'_2) mit $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$.
   $(p_1, p_2) \xrightarrow{\alpha}_{12} (q_1, q_2)$ kann in $P_1 \| P_2$ für ein α aus $\text{Synch}(P_1, P_2)$ nur gelten, wenn in P_1 die α-Transition zwischen p_1 und q_1 auch bereits eine must-Transition war und analog für p_2 und q_2 in P_2. Somit erzwingen die $\mathcal{R}_j$ für die Komponenten bereits die Implementierung der must-Transitionen, so dass es dann entsprechende $(q_j, q'_j) \in \mathcal{R}_j$ gibt. Die Parallelkomposition der implementierten must-Transitionen aus P'_1 und P'_2 führt in $P'_1 \| P'_2$ zu der geforderten Transition. Falls α keine synchronisiert Aktion ist, enthält die Parallelkomposition die Transition nur, da eine Komponente diese Transition alleine ausführen kann (ersten beiden Zeilen der $\longrightarrow_{12}$ Definition in 1.2). OBdA $p_1 \xrightarrow{\alpha}_1 q_1$ und somit gilt $p_2 = q_2$. Da $\mathcal{R}_1$ eine as-Verfeinerungs-Relation ist, gibt es in P'_1 zwischen p'_1 und q'_1 eine must-Transition und es gilt $(q_1, q'_1) \in \mathcal{R}_1$. α ist auch in der Parallelkomposition der as-Implementierungen eine unsynchronisierte Aktion und somit entsteht die Transition dort auch nur aus der Transition von P'_1 und es gilt $p'_2 = q'_2$.
   Da in beiden Fällen $(q_j, q'_j) \in \mathcal{R}_j$ für beide Werte von i gilt, gilt nach der Definition von $\mathcal{R}_{12}$ auch $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$. $p_2 \mathcal{R}_2 p'_2$ musst nach Voraussetzung gelten und somit gilt wegen der Gleichheiten der Zustände auch $q_2 \mathcal{R}_2 q'_2$.

2. Es ist $(p'_1, p'_2) \overset{\alpha}{\dashrightarrow}_{P'_1 \| P'_2} (q'_1, q'_2)$ impliziert $(p_1, p_2) \overset{\alpha}{\dashrightarrow}_{12} (q_1, q_2)$ für ein (q_1, q_2) mit $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$ für diesen Punkt zu zeigen.
   Die Argumentation könnte wie in Punkt 1. erneut in unsynchronisierte und synchronisierte Aktionen gesplittet werden. Jedoch würde man dadurch nur auf das Ergebnis kommen, dass die eine Komponente oder beide die entsprechende may-Transition ausführen können müssen, damit die Parallelkomposition dies auch kann. Es gilt also mindestens für eine der Komponenten $p'_j \overset{\alpha}{\dashrightarrow}_{P'_j} q'_j$ und durch die Definition 1.3, die für die entsprechende Relation $\mathcal{R}_j$ gilt, muss es in P_j die Transition $p_j \overset{\alpha}{\dashrightarrow}_j q_j$ geben, so dass dann $(q_j, q'_j) \in \mathcal{R}_j$ gilt. Durch die Definition von $\dashrightarrow$ in 1.2 werden die may-Transitionen der Komponenten entsprechend in die Parallelkomposition $P_1 \| P_2$ aufgenommen und die Relation $\mathcal{R}_{12}$ gilt, mit den analogen Begründungen wie in 1., auch für die entsprechenden Zustands-Tupel.

3. Hierfür muss gezeigt werden, wenn $(p'_1, p'_2) \in E_{P'_1 \| P'_2}$ gilt, dann ist auch (p_1, p_2) in $P_1 \| P_2$ ein Zustand, der in der Menge der Fehler-Zustande E_{12} enthalten ist.
   Falls (p'_1, p'_2) ein geerbter Fehler ist, dann ist oBdA $p'_1 \in E_{P'_1}$ und $p'_2 \in P'_2$. Aufgrund von $\mathcal{R}_1$ und Definition 1.3 3. muss dann auch $p_1 \in E_1$ gelten. Für p_2 gilt durch die Signatur von $\mathcal{R}_2$ $p_2 \in P_2$. Zusammen in $P_1 \| P_2$ ergibt das wieder einen geerbten Fehler, also $(p_1, p_2) \in E_{12}$. (p'_1, p'_2) kann jedoch auch ein neuer Kommunikationsfehler sein, dann gilt oBdA $p'_1 \overset{a}{\not\longrightarrow}_{P'_1}$ und $p'_2 \overset{a}{\dashrightarrow}_{P'_2}$ für ein a aus $I_1 \cap O_2$. Aufgrund von Definition 1.3 2. muss dann auch $p_2 \overset{a}{\dashrightarrow}_2$ gelten. Für p_1 kann nicht $p_1 \overset{a}{\longrightarrow}_1$ gelten, da sonst die Simulations Relation $\mathcal{R}_1$ die Implementierung dieser Transition in P'_1 fordern würde (1.3 1.). Es gilt also $p_1 \overset{a}{\not\longrightarrow}_1$ und in der Parallelkomposition $P_1 \| P_2$ ergibt sich daraus ebenfalls ein neuer Kommunikationsfehler mit $(p_1, p_2) \in E_{12}$.

$\Rightarrow (P'_1 \| P'_2) \in \text{as-impl}(P_1 \| P_2)$. □

Die entgegengesetzte Richtung von Lemma 2.3 gilt im allgemeinen nicht, d.h. es muss zu einer as-Implementierung einer Parallelkomposition $P' \in \text{as-impl}(P_1 \| P_2)$ keine as-Implementierungen P'_1 bzw. P'_2 der einzelnen Komponenten P_1 bzw. P_2 geben, deren Parallelkomposition $P'_1 \| P'_2$ der as-Implementierung 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-Implementierung unabhängig entschieden werden kann, ob sie implementiert werden oder nicht. Somit kommt es in P' zu dem Problem, dass keine as-Implementierung von P_2 (entweder keine Transition implementiert oder die o' Transition ist implementiert) in Parallelkomposition mit der Implementierung P_1 P' ergeben würde.

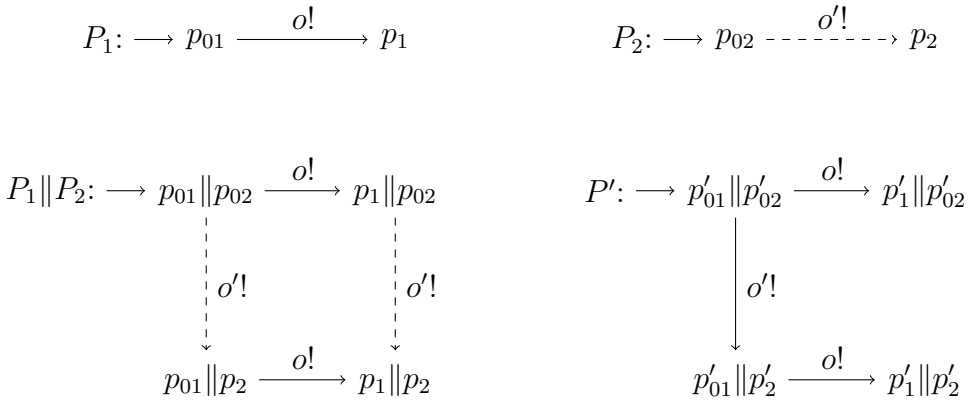

Abbildung 2.1: Gegenbeispiel für Umkehrung von Lemma 2.3

Lemma 2.4 (*w-as-Implementierungen und Parallelkomposition*).
$P'_1 \in \text{w-as-impl}(P_1) \wedge P'_2 \in \text{w-as-impl}(P_2) \Rightarrow (P'_1 \| P'_2) \in \text{w-as-impl}(P_1 \| P_2)$.

Beweis. Es gelte $j \in \{1,2\}$. Da $P'_j \in \text{w-as-impl}(P_j)$ gilt, gibt es nach Definition 1.4 eine schwache as-Verfeinerungs-Relation $\mathcal{R}_j$, die beschreibt, wie P'_j P_j verfeinert. Analog zum Beweis von Lemma 2.3 kann auch hier die neue schwache as-Verfeinerungs-Relation der Parallelkompositionen $\mathcal{R}_{12}$ auf Basis der $\mathcal{R}_j$ definiert werden: $\forall p'_1, p'_2, p_1, p_2 : ((p'_1, p'_2), (p_1, p_2)) \in \mathcal{R}_{12} \Leftrightarrow (p'_1, p_1) \in \mathcal{R}_1 \wedge (p'_2, p_2) \in \mathcal{R}_2$. Es bleibt nun zu zeigen, dass $\mathcal{R}_{12}$ eine zulässige schwache Verfeinerungs-Relation nach Definition 1.4 ist, da die Parallelkomposition von zwei Implementierungen auch immer eine Implementierung ist (1.2).

1. Aus der schwache Simulations-Definition 1.4 folgt, dass für den ersten Punkt folgendes zu zeigen ist: $(p_1, p_2) \xrightarrow{i}_{12} (q_1, q_2)$ impliziert $(p'_1, p'_2) \xrightarrow{i}_{P'_1 \| P'_2} \stackrel{\varepsilon}{\Longrightarrow}_{P'_1 \| P'_2} (q'_1, q'_2)$ für ein (q'_1, q'_2) mit $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$.
   Falls die Transition $(p_1, p_2) \xrightarrow{i}_{12} (q_1, q_2)$ in der Parallelkomposition $P_1 \| P_2$ für ein $i \in \text{Synch}(P_1, P_2)$ möglich ist, dann gab es diese Transition auch bereits in P_1 und P_2 als must-Transition. Somit verlangen bereits die schwachen as-Verfeinerungs-Relation $\mathcal{R}_1$ und $\mathcal{R}_2$, dass $p'_j \xrightarrow{i}_{P'_j} \stackrel{\varepsilon}{\Longrightarrow}_{P'_j} q'_j$ für $j = 1$ und $j = 2$ gilt. Wenn auf die MEIOs mit diesen Transitionen die Parallelkomposition angewendet wird, entstehen der Ablauf $(p'_1, p'_2) \xrightarrow{i}_{P'_1 \| P'_2} \stackrel{\varepsilon}{\Longrightarrow}_{12} (q'_1, q'_2)$. Die q'_j müssen in der Relation $\mathcal{R}_j$ mit dem jeweiligen q_j ein Tupel bilden. Somit gilt auch $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$. Falls $i \notin \text{Synch}(P_1, P_2)$, ist die i Transition in $P_1 \| P_2$ nur aufgrund einer entsprechenden Transition in einer Komponente möglich. OBdA gilt $p_1 \xrightarrow{i}_1 q_1$ und mit Definition 1.4 1. gilt dann $p'_1 \xrightarrow{i}_{P'_1} q'_1$ und $(q_1, q'_1) \in \mathcal{R}_1$. Da es sich um eine unsynchronisierte Aktion von P_1 bzw. P'_1 handelt, muss für die zweite Komponente $(p_2, q_2) = (p'_2, q'_2) \in \mathcal{R}_2$ gelten. Die interne Aktion ist immer unsynchronisiert und i in diesem Fall auch, deshalb ist $(p'_1, p'_2) \xrightarrow{i}_{P'_1 \| P'_2} \stackrel{\varepsilon}{\Longrightarrow}_{P'_1 \| P'_2} (q'_1, q'_2)$ die in der Parallelkomposition entstehende Transitionsfolge. $(q_1, q_2) \mathcal{R}_{12} (q'_1, q'_2)$ gilt für den Fall des unsynchronisierten i's ebenfalls.

2. Analog zu 1. kann für diesen Punkt $(p_1, p_2) \xrightarrow{\omega}_{12} (q_1, q_2)$ impliziert $(p'_1, p'_2) \stackrel{\hat{\omega}}{\Longrightarrow}_{P'_1 \| P'_2} (q'_1, q'_2)$ für ein (q'_1, q'_2) mit $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$ gezeigt werden.
   Die ω Transition in $P_1 \| P_2$ ist entweder aus einem synchronisierten oder aus einem unsynchronisierten ω entstanden. Entsprechend ist dann in einem oder beiden P_j die Transitionen möglich und durch die Relationen $\mathcal{R}_j$ folgen die Transitionen $p'_j \stackrel{\omega}{\Longrightarrow}_{P'_j} q'_j$ mit $(q_j, q'_j) \in \mathcal{R}_j$. Durch die Parallelkomposition von P'_1 mit P'_2 folgt dann das zu zeigende.

3. $(p'_1, p'_2) \xrightarrow{i}_{P'_1 \| P'_2} (q'_1, q'_2)$ impliziert $(p_1, p_2) \xrightarrow{i}_{12} \stackrel{\varepsilon}{\Longrightarrow}_{12} (q_1, q_2)$ für ein (q_1, q_2) mit $((q_1, q_2), (q'_1, q'_2)) \in \mathcal{R}_{12}$ ist die Voraussetzung des 3. 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 Komponenten und $i \notin \text{Synch}(P'_1 \| P'_2)$. In beiden Fällen gilt für ein j oder beide $p'_j \xrightarrow{i}_{P'_j} q'_j$. Daraus folgt mit $\mathcal{R}_j$ $p_j \xrightarrow{i}_j \stackrel{\varepsilon}{\Longrightarrow}_j q_j$ mit (q_j, q'_j). Durch Syn-

chronisation oder durch übernehmen der Transition folgt dann $(p_1, p_2) \overset{i}{\dashrightarrow}_{12} \overset{\varepsilon}{\Longrightarrow}_{12} (q_1, q_2)$ und mit Hilfe der Definition von $\mathcal{R}_{12}$ auch $((q_1, q_2), (q_1', q_2')) \in \mathcal{R}_{12}$.

4. Analog zu 3. kann für diesen Punkt $(p_1', p_2') \overset{\omega}{\dashrightarrow}_{P_1' \| P_2'} (q_1', q_2')$ impliziert $(p_1, p_2) \overset{\hat{\omega}}{\Longrightarrow}_{12} (q_1, q_2)$ 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. Entsprechend ist dann in einem oder beiden P_j' die Transitionen möglich und durch die Relationen $\mathcal{R}_j$ folgen die Transitionen $p_j \overset{\omega}{\Longrightarrow}_j q_j$ mit $(q_j, q_j') \in \mathcal{R}_j$. Durch die Parallelkomposition von P_1 mit P_2 folgt dann das zu zeigende.

5. Für diesen Punkt ist zu zeigen, dass aus $(p_1', p_2') \in E_{P_1' \| P_2'}$ $(p_1, p_2) \in E_{12}$ folgt.
   Falls $(p_1', p_2') \in E_{P_1' \| P_2'}$ ein geerbter Fehler ist, gilt oBdA $p_1' \in E_{P_1'}$ und $p_2' \in P_2'$. Mit Definition 1.4 5., die für $\mathcal{R}_1$ gilt, folgt $p_1 \in E_1$. p_2 steht mit p_2' in der Relation $\mathcal{R}_2$. Durch die Signatur folgt $p_2 \in P_2$. Es gilt also in der Parallelkomposition $P_1 \| P_2$ $(p_1, p_2) \in E_{12}$ ist ein von P_1 geerbter Fehler. (p_1', p_2') kann jedoch in $P_1' \| P_2'$ auch ein neuer Kommunikationsfehler sein. Dann gilt oBdA $p_1' \overset{a}{\not\dashrightarrow}_{P_1'}$ und $p_2' \overset{a}{\dashrightarrow}_{P_2'}$ für ein $a \in I_0 \cap O_2$. Mit $\mathcal{R}_2$ und 1.4 4. folgt die Gültigkeit von $p_2 \overset{a}{\Longrightarrow}_2$ in P_2. Da es für p_1' keine ausgehende a must-Transition geben darf, darf es auch in P_1 keine a must-Transition von p_1 aus geben, ansonsten wäre 1.4 1. für $\mathcal{R}_1$ verletzt. (q_1, q_2) ist also auch ein neuer Kommunikationsfehler und somit in E_{12} enthalten.

$\Rightarrow (P_1' \| P_2') \in \text{w-as-impl}(P_1 \| P_2)$. □

Ein neuer Kommunikationsfehler in einer Parallelkomposition muss in einer Implementierung (as oder w-as) nicht auftauchen, auch nicht in der Parallelkomposition von Implementierungen der einzelnen Komponenten. Dies liegt daran, dass für den Input nur gesagt 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 Kommunikationsfehler kommt, dann gibt es auch immer mindestens eine Implementierung, die diesen Kommunikationsfehler enthält und es gibt auch immer mindestens ein Implementierungs-Paar der Komponenten, in deren Parallelkomposition sich dieser Kommunikationsfehler ebenfalls zeigt.

3 Verfeinerungen für Kommunikationsfehler-Freiheit

Dieses Kapitel versucht die Präkongruenz für Error bei EIOs aus [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 ein Fehler-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 dieser Fehler-Zustand auch lokal erreichbar ist. Die Basisrelation stellt eine* Verfeinerung *bezüglich* Kommunikationsfehlern *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 in [Sch16] werden die Fehler hier Trace-basiert betrachtet. Da jedoch die Basisrelation über as-Implementierungen spricht, sind die Trace-Mengen auch nicht für die MEIOs mit may-Transitionen definiert sondern nur für die Menge der möglichen as-Implementierungen eines solchen MEIOs.

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

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

Proposition 3.4 (*Kommunikationsfehler-Traces und Implementierung*). *Für ein MEIO P gilt:*

1. strikte Kommunikationsfehler-Traces*:* $StET(P) = \left\{ w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p_0' \overset{w}{\Longrightarrow}_{P'} p' \in E \right\}$ TODO: erzwungen Zeilenumbruch kontrollieren

2. Input-kritische-Traces: $MIT(P) = \left\{ wa \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \wedge a \in I \wedge p' \stackrel{a}{\not\rightarrow}_{P'} \right\}$ TODO: erzwungen Zeilenumbruch kontrollieren

Beweis.

1. Wie schon in Beweis zu 2.1 festgestellt, sind alle Abläufe, die in P via may-Transitionen möglich ist in mindestens einer as-Implementierung von P via must-Transitionen möglich. Umgekehrt ist auch jeder Ablauf, der in einer as-Implementierung von P möglich ist auch in P durch may-Transitionen möglich.
   Aufgrund des 3. Punktes der Definition 1.3 kann jede as-Implementierung von P nur Fehler-Zustände enthalten, die auch P enthält. Da alle möglichen Implementierungen von P in $\left\{ w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \in E \right\}$ betrachtet werden, ist auch jeder in P durch may-Transitionen erreichbare Fehler-Zustand auch in mindestens einer as-Implementierung ebenfalls erreichbar, jedoch durch must-Transitionen.

2. Für jedes w in $L(P)$ gibt es mindestens eine as-Implementierung von P, die dieses w auch ausführen kann und umgekehrt (Beweis von 2.1). Falls in $MIT(P)$ mindestens ein Element gibt, gibt es in P einen Trace von w, nach dem ein Input a nicht zwingendermaßen folgen muss. Die a Transition also entweder eine may-Transition ist oder gar nicht existiert in P. Es muss also auch einen w Trace in einer as-Implementierung geben, die in einem Zustand endet, der mit dem Zustand aus P in Relation steht, in dem das a nicht erzwungen wird. Falls die Transition in P nicht vorhanden ist, muss jede as-Implementierung, die so einen w Trace enthält auch wa als Input-kritischen-Trace haben. Andernfalls handelt es sich bei a in P um eine may-Transition, die von mindestens einer as-Implementierung, die den w Trace enthält, nicht implementiert wird und somit wa auch als Input-kritischen-Trace enthält.
   Für ein $wa \in \left\{ wa \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \wedge a \in I \wedge p' \stackrel{a}{\not\rightarrow}_{P'} \right\}$ muss es eine as-Implementierung von P geben, die diesen Input-kritischen-Trace implementiert. P muss also auch das w ausführen können zu einem Zustand, in dem P a nicht als must-Transition enthalten. Falls P a nach w nur als must-Transition enthalten würde, würde 1.3 1. die Implementierung von a erzwingen würde und somit könnte für keine as-Implementierung von P wa ein Input-kritischer-Trace sein. Es gilt also auch $wa \in MIT(P)$.

□

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

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

Satz 3.6 (*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 Errortraces 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 $(p_{01}, p_{02}) \stackrel{x}{\Rightarrow}_{12} (p_1, p_2) \wedge a \in I_{12} \wedge (p_1, p_2) \stackrel{a}{\not\rightarrow}_{12}$. 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 $p_{01} \stackrel{x_1}{\Rightarrow}_1 p_1 \stackrel{a}{\not\rightarrow}_1$ und $p_{02} \stackrel{x_2}{\Rightarrow}_2 p_2 \stackrel{a}{\not\rightarrow}_2$ oder $p_{02} \stackrel{x_2}{\Rightarrow}_2 p_2 \stackrel{a}{\rightarrow}_2$ 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: $p_{01} \stackrel{x_1}{\Rightarrow}_1 p_1 \stackrel{a}{\not\rightarrow}_1$ und $p_{02} \stackrel{x_2}{\Rightarrow}_2 p_2$ 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 $v \in O_{12}^*$ gilt, so dass $(p_{01}, p_{02}) \stackrel{w}{\Rightarrow}_{12} (p_1, p_2) \stackrel{v}{\Rightarrow}_{12} (p_1', p_2')$ gilt mit $(p_1', p_2') \in E_{12}$ und $w = \text{prune}(wv)$. Durch Projektion auf die Komponenten erhält man $p_{01} \stackrel{w_1}{\Rightarrow}_1 p_1 \stackrel{v_1}{\Rightarrow}_1 p_1'$ und $p_{02} \stackrel{w_2}{\Rightarrow}_2 p_2 \stackrel{v_2}{\Rightarrow}_2 p_2'$ 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 Kommunikationsfehler hingegen wäre durch das fehlen der Synchronisations-Erzwingung (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 $p_{02} \stackrel{w_2 v_2}{\Rightarrow}_2$ 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 Kommunikationsfehler): OBdA gilt $a \in I_1 \cap O_2$ mit $p_1' \overset{a}{\not\rightarrow}_1 \wedge p_2' \overset{a}{\dashrightarrow}_1$. 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 $p_{01} \overset{v_1'}{\Longrightarrow}_1 p_1 \overset{a}{\not\rightarrow}_1$ und es gilt $v' = v'' a$.

  – Fall Ia) ($a \notin \Sigma_2$): Es gilt $p_{02} \overset{v_2'}{\Longrightarrow}_2 p_2$ mit $v'' \in v_1' \| v_2'$. Dadurch erhält man $(p_{01}, p_{02}) \overset{v''}{\Longrightarrow}_{12} (p_1, p_2) \overset{a}{\not\rightarrow}_{12}$ 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 $p_{02} \stackrel{v_2''}{\Rightarrow}_2 \stackrel{a}{\not\rightarrow}_2$ und $v'' \in v_1' \| v_2''$. a ist für P_2, ebenso wie für P_1, ein nicht erzwungener Input. Daraus folgt, dass $(p_1, p_2) \stackrel{a}{\not\rightarrow}_{12}$ 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 $p_{02} \stackrel{v_2''}{\Rightarrow}_2 p_2 \stackrel{a}{\longrightarrow}_2$ mit $v_2' = v_2''a$. Da die gemeinsamen Inputs synchronisiert werden, folgt $(p_1, p_2) \stackrel{a}{\not\rightarrow}_{12}$ bereits aus $q_1 \stackrel{a}{\not\rightarrow}_1$. 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 $p_{02} \stackrel{v_2'}{\Rightarrow}_2$. Man erhält also $p_{02} \stackrel{v_2''}{\Rightarrow}_2 \stackrel{a}{\dashrightarrow}_2$ mit $v'' \in v_1' \| v_2'$. Daraus ergibt sich $(p_{01}, p_{02}) \stackrel{v''}{\Rightarrow}_{12} (p_1, p_2)$ mit $p_1 \stackrel{a}{\not\rightarrow}_1, a \in I_1, q_2 \stackrel{a}{\dashrightarrow}_1$ 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 ($v_1 \in PrET_1$): $\exists u_1 \in O_1^* : p_{01} \stackrel{v_1}{\Rightarrow}_1 p_1 \stackrel{u_1}{\Rightarrow}_1 p_1'$ 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 $p_{02} \stackrel{v_2'}{\Rightarrow}_2 p_2$ mit $(p_{01}, p_{02}) \stackrel{v'}{\Rightarrow}_{12} (q_1, q_2)$.

  - 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 $p_2 \stackrel{u_2}{\Longrightarrow}_2 p_2' \stackrel{c}{\not\rightarrow}_2\Big)$: Für das Präfix $u_1'c$ von u_1 mit $u_1'c|_{I_2} = u_2c$ weiß man, dass $q_1 \stackrel{u_1'}{\Rightarrow}_1 q_1'' \stackrel{c}{\dashrightarrow}_1$. Somit gilt $u_1' \in u_1' \| u_2$ und $(p_1, p_2) \stackrel{u_1'}{\Rightarrow}_{12} (q_1'', q_2') \in E_{12}$, der für P_2 der entsprechende Input nicht erzwungen wird, der mit dem c Output von P_1 zu koppeln wäre. Es handelt sich also um einen neuen Kommunikationsfehler. 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) $\Big(p_2 \stackrel{u_2}{\Longrightarrow}_2 p_2'$ mit $u_2 = u_1|_{I_2}\Big)$: Es gilt $u_1 \in u_1 \| u_2$ und $(p_1, p_2) \stackrel{u_1}{\Rightarrow}_{12} (p_1', p_2') \in E_{12}$, 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:

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

□

Korollar 3.7 (*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:

- $$\begin{aligned}ET_{31} &\overset{3.6\ 1.}{=} \operatorname{cont}(\operatorname{prune}((ET_3 \| EL_1) \cup (EL_3 \| ET_1))) \\ &\overset{\substack{ET_1 \subseteq ET_2 \\ \text{und} \\ EL_1 \subseteq EL_2}}{\subseteq} \operatorname{cont}(\operatorname{prune}((ET_3 \| EL_2) \cup (EL_3 \| ET_2))) \\ &\overset{3.6\ 1.}{=} ET_{32},\end{aligned}$$
- $$\begin{aligned}EL_{31} &\overset{3.6\ 2.}{=} (EL_3 \| EL_1) \cup E_{31} \\ &\overset{\substack{EL_1 \subseteq EL_2 \\ \text{und} \\ ET_{31} \subseteq ET_{32}}}{\subseteq} (EL_3 \| EL_2) \cup ET_{32} \\ &\overset{3.6\ 2.}{=} EL_{32}.\end{aligned}$$

□

Lemma 3.8 (*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):
  - $U = \{p_0, p_1, \dots, p_{n+1}\}$,
  - $p_{0U} = p_0$,
  - $\longrightarrow_U = \{(p_i, x_{i+1}, p_{i+1}) \mid 0 \leq i \leq n\}$
    $\cup \{(p_i, x, p_{n+1}) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\}$
    $\cup \{(p_{n+1}, x, p_{n+1}) \mid x \in I_U\}$.
  - $E_U = \emptyset$,

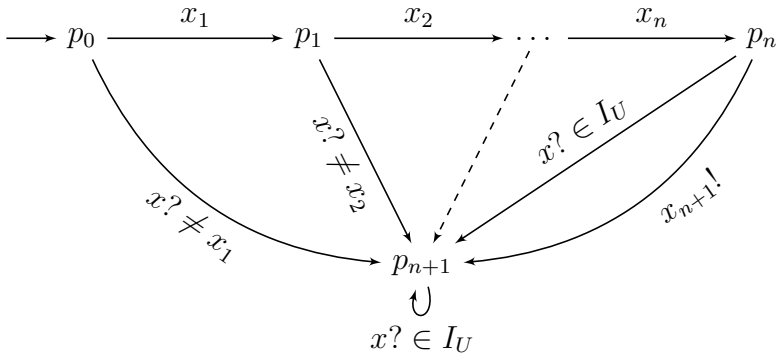

Abbildung 3.1: $x? \neq x_i$ steht für alle $x \in I_U \backslash \{x_i\}$

Für w können nun zwei Fälle unterschieden werden. Aus beiden wird folgen, dass für mindestens eine as-Implementierung P' von $U \| P_1$ $\varepsilon \in PrET(P')$.

– Fall 2a) ($w \in MIT_1$): In $U\|P_1$ erhält man $(p_1, p_{01}) \stackrel{x_1 \ldots x_n}{=\!=\!\Rightarrow}_{U\|P_1} (p_n, p')$ mit $p' \stackrel{x_{n+1}}{\not\longrightarrow}_1$ und $p_n \stackrel{x_{n+1}}{\longrightarrow}_U$. Deshalb gilt $(p_1, p') \in E_{U\|P_1}$. Da alle Aktionen aus w bis auf x_{n+1} synchronisiert werden und $I \cap I_U = \emptyset$, gilt $x_1, \ldots x_n \in O_{U\|P_1}$. Da $(p_1, p') \in E_{U\|P_1}$ gibt es mindestens ein P' in as-impl($U\|P_1$), die diesen Fehler-Zustand ebenfalls enthält. Daraus ergibt sich dann $\varepsilon \in PrET(P')$.

– Fall 2b) ($w \in PrET_1$): In $U\|P_1$ erhält man $(p_0, p_{01}) \stackrel{w}{=\!=\!\Rightarrow}_{U\|P_1} (p_{n+1}, p'') \stackrel{u}{=\!=\!\Rightarrow}_{U\|P_1}$ (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 $x_1, \ldots, x_n, x_{n+1} \in O_{U\|P_1}$ und, da $u \in O^*$, folgt $u \in O^*_{U\|P_1}$. Somit ergibt sich für eine as-Implementierung 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 jeder in einer as-Implementierung von $U\|P_2$ lokal erreichbare Fehler auch in $U\|P_2$ lokal erreichbar sein.
Dieser 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 erzwingt, 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 Kommunikationsfehler muss also die Form (p_i, p') haben mit $i \leq n, p' \stackrel{x_{i+1}}{\not\longrightarrow}_2$ und $x_{i+1} \in O_U = I$. Durch Projektion erhält man dann $p_{02} \stackrel{x_1 \ldots x_i}{=\!=\!\Rightarrow}_2 p' \stackrel{x_{i+1}}{\not\longrightarrow}_2$ und damit gilt $x_1 \ldots x_{i+1} \in MIT_2 \subseteq ET_2$. Somit ist ein Präfix von w in ET_2 enthalten.

– Fall 2ii) (geerbter Fehler): U hat $x_1 \ldots x_i u$ mit $u \in I_U^* = O^*$ ausgeführt und ebenso hat P_2 dieses Wort abgearbeitet. Durch dies hat P_2 einen Zustand E_2 erreicht, da von U kleine Fehler geerbt werden können. Es gilt dann prune($x_1 \ldots x_i u$) = prune($x_1 \ldots x_i$) $\in PrET_2 \subseteq ET_2$. 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 Kommunikationsfehler-Traces entspricht, ist $x_1 \ldots x_i$ 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 nicht gezeigt werden.
- Fall 2 ($w = x_1 \ldots x_n$ mit $n \geq 1$): Es wird ein Partner U wir folgt konstruiert (siehe dazu auch Abbildung 3.2)

- $U = \{p_0, p_1, \ldots, p_n, p\}$,
- $p_{0U} = p_0$,
- $\longrightarrow_U = \{(p_i, x_{i+1}, p_{i+1}) \mid 0 \leq i < n\}$
  $\cup \{(p_i, x, p) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i < n\}$
  $\cup \{(p, x, p) \mid x \in I_U\}$.
- $E_U = \{p_n\}$,

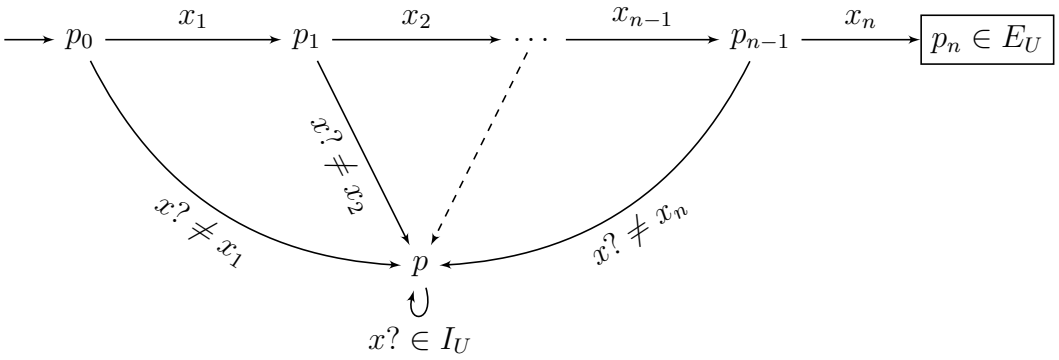

Abbildung 3.2: $x? \neq x_i$ steht für alle $x \in I_U \backslash \{x_i\}$, p_n ist der einzige Fehler-Zustand

Da $p_{01} \stackrel{w}{\Longrightarrow}_1 p'$ gilt, kann man schließen, dass $U \| P_1$ einen lokal erreichbaren geerbten Fehler hat. Es gibt also auch mindestens eine Implementierung in 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 dieser Fehler-Zustand auch in $U \| P_2$ lokal erreichbar sein.

- Fall 2a) (neuer Fehler aufgrund von $x_i \in O_U$ und $p_{02} \stackrel{x_1 \ldots x_{i-1}}{\Longrightarrow}_2 q'' \not\stackrel{x_i}{\longrightarrow}_2$): Es gilt $x_1 \ldots x_i \in MIT_2$ 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 Input 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 $p_{02} \stackrel{x_1 \ldots x_n}{\Longrightarrow}_2$ gilt. In diesem Fall gilt $w \in L_2 \subseteq EL_2$.

– Fall 2d) (geerbter Fehler von P_2): Es gilt dann $p_{02} \stackrel{x_1 \dots x_i u}{=\!=\!\Rightarrow}_2 p' \in E_2$ für $i \geq 0$ und $u \in O^*$. Somit ist $x_1 \dots x_i u \in StET_2$ und damit $\text{prune}(x_1 \dots x_i u) = \text{prune}(x_1 \dots x_i) \in PrET_2 \subseteq EL_2$. Somit gilt $w \in EL_2$.

□

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 3.9 (*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 ins 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 Erregbarkeit der Fehler-Zustande in den as-Implementierungen definiert wurde und die ET-Mengen von P_1 und P_2 auch durch die Vereinigung der Traces ihrer as-Implementierungen, wie in Proposition 3.4, ausgedrückt werden können. Es ist also $\sqsubseteq_E$ in $\sqsubseteq_E^{\mathrm{B}}$ enthalten. Wie in Korollar 3.7 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 3.8 folgt dann $P_1 \sqsubseteq_E P_2$. □

Es wurde somit jetzt 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 fehler-frei mit U kommuniziert. Dann wird auch diese Verfeinerung durch $\sqsubseteq_E$ charakterisiert.

Korollar 3.10. *Es gilt: $P_1 \sqsubseteq_E P_2 \Leftrightarrow U \| P_1 \sqsubseteq_E^{\mathrm{B}} U \| P_2$ für alle Partner U.*

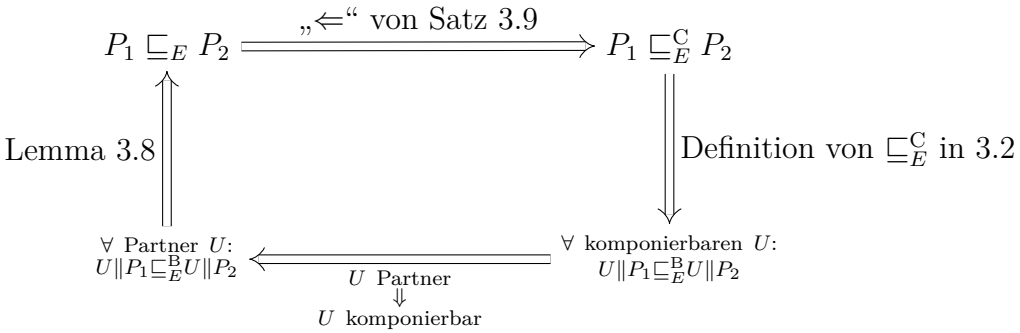

Abbildung 3.3: Folgerungskette der Kommunikationsfehler-Relationen

4 Verfeinerungen für Kommunikationsfehler- und Ruhe-Freiheit

In diesem Kapitel wird die Menge der betrachteten Zustandsmengen von den Kommunikationsbasierten Fehlern im letzten Kapitel erweitert um Ruhe-Zustände.
Zustände, die keine Outputs ohne einen Input ausführen können in Form einer must-Transition, werden als in einer Art Verklemmung angesehen, da sie ohne Zutun von Außen den Zustand nicht mehr verlassen können, falls 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 erzwungen 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 weider 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 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: $Qui(P) := \left\{p \in P \mid \forall \alpha \in (O \cup \{\tau\}) : p \overset{\alpha}{\not\longrightarrow}_P\right\}$.

Proposition 4.2 (*Ruhe und Implementierung*). *Für ein MEIO P gilt:* $Qui(P) = \left\{p' \in P' \mid P' \in \text{as-impl}(P) \wedge \forall \alpha \in (O \cup \{\tau\}) : p' \overset{\alpha}{\not\longrightarrow}_{P'}\right\}$.

Beweis. Da für alle $P' \in \text{as-impl}(P)$ $\longrightarrow_{P'} = \dashrightarrow_{P'}$ gilt, dürfen alle p' keine ausgehenden Transitionen haben. Nach Definition 1.3 dürfen zu p' in Relation stehenden Zustände p in P keine ausgehenden must-Transitionen haben, da p' diese sonst implementieren müsste. Somit sind alle p', die die Forderung dieser Proposition enthalten auch in $Qui(P)$ enthalten. Für alle p, die in $Qui(P)$ enthalten sind, gibt es keine ausgehenden must-Transitionen für Outputs oder die interne Aktion τ, somit können diese Aktionen entweder keine ausgehende Transition von p sein oder nur als may-Transition von p zu einem möglicherweise anderen Zustand führen. Nicht vorhandenen Transitionen in P sind auch in keiner as-Implementierung von P enthalten und es gibt mindestens eine Implementierung in as-impl(P), die alle ausgehenden may-Transitionen der lokalen Aktionen von p nicht implementiert, somit sind alle p aus $Qui(P)$ auch in $\left\{p' \in P' \mid P' \in \text{as-impl}(P) \wedge \forall \alpha \in (O \cup \{\tau\}) : p' \overset{\alpha}{\not\longrightarrow}_{P'}\right\}$ enthalten. □

Für die Erreichbarkeit wird wie im letzten Kapitel ein optimistischer Anzahl der lokalen Erreichbarkeit für die Fehler-Zustände, der Kommunikationsfehler, verwendet. Ruhe ist kein unabwendbarer Fehler, sondern kann durch einen Input repariert werden oder im Fall von vorhandenen may-Output-Transitionen, durch eine Implementierung dieses Outputs. Daraus ergibt sich, dass Ruhe im Vergleich zu Kommunikationsfehler 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 4.3 (*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.*

Definition 4.4 (*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 solcher lokal erreichbar ist. Diese* Basisrelation *stellt eine* Verfeinerung *bezüglich* Kommunikationsfehlern *und* Ruhe *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 reparierbarer Fehler im Gegensatz zu Kommunikationsfehlern. Es genügt deshalb für Ruhe die strikten Traces ohne Kürzung zu betrachten.

Definition 4.5 (*Ruhe-Traces*). *Sei P ein MEIO und definiere:*

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

Proposition 4.6 (*Ruhe-Traces und Implementierung*). *Für ein MEIO P gilt* $StQT(P) = \left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \in Qui(P')\right\}$.

Beweis. Da P' einen mit w beschrifteten must-Trace enthält und P' eine as-Implementierung von P ist, muss w als may-Trace bereits in P möglich gewesen sein (Definition 1.3). Der Zustand p' kann in P' auch nur ruhig sein, wenn der entsprechende in Relation stehende Zustand aus P auch bereits ruhig war. Alle w aus $\left\{w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \in Qui(P')\right\}$ sind somit also auch in $StQT(P)$ enthalten. Jeder may-Trace aus P kann implementiert werden durch eine as-Implementierung. Die Menge as-impl(P) enthält alle as-Implementierungen von P, also auch eine, die den w may-Trace aus $StQT(P)$ zu einem mit in p Relation stehenden Zustand implementiert. p enthält keine ausgehenden Transitionen für lokale Aktionen, also gibt es unter den as-Implementierungen, die w entsprechend als must-Trace enthalten auch mindestens eine, die keine ausgehende lokale Aktion an dem zu p in Relation stehenden Zustand implementiert. Die Behauptung gilt also. □

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

Definition 4.7 (*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.

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.*

   TODO: anpassen (nur einer muss ruhig sein, der andere kann must-Transitionen für synchronisierte Outputs haben)

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 $\left((p_1, p_2) \xrightarrow{\tau}\right)$: 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 für einen der beiden Zustände als must-Transition im der einzelnen 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 $\left((p_1, p_2) \xrightarrow{a} \text{ mit } a \in O_{12} \backslash \text{Synch}(P_1, P_2)\right)$: 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 $p_1 \xrightarrow{a}$ 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 $\left((p_1, p_2) \xrightarrow{a} \text{ mit } a \in O \cap \text{Synch}(P_1, P_2)\right)$: 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 $p_1 \xrightarrow{a}$ und $p_2 \xrightarrow{a}$. 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.

2. TODO: zu beweisen

□

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.
TODO: zu beweisen □

5 Verfeinerungen für Kommunikationsfehler-, Ruhe- und Divergenz-Freiheit

In diesem Kapitel soll die Menge der betrachteten Zustände noch einmal erweitert werden. Somit werden dann Fehler-, Ruhe- und Divergenz-Zustände betrachtet.

Definition 5.1 (*Divergenz*). *Ein* Divergenz-Zustand *ist ein Zustand in einem MEIO P, der eine unendliche Folge an τs ausführen kann via may-Transitionen.*
Die Menge $Div(P)$ besteht aus all diesen divergenten Zuständen des MEIOs P.

Proposition 5.2 (*Divergenz und Implementierung*). *Für ein MEIO P gilt $Div(P) = \{p' \in P' \mid P' \in \text{as-impl}(P) \wedge P'$ kann von p' aus eine unendliche Folge von τs ausführen$\}$.*

Beweis. Falls von einem p' in einem $P' \in \text{as-impl}(P)$ eine unendliche τ Folge ausführbar ist, dann ist dies via must-Transitionen möglich. Diese Transitionen müssen jedoch aus möglichen may-Transitionen in P folgen. Deshalb war auch der Zustand, der zu p' in Simulations-Relation nach Definition 1.3 steht, auch bereits divergent und in $Div(P)$ enthalten. Für jedes p aus $Div(P)$ gibt es eine unendliche Folge von τs, die via may-Transitionen möglich ist in P. Also gibt es auch in as-impl(P) mindestens eine Implementierung, die alle diese beteiligen may-Transitionen als must-Transitionen implementiert und einen zu p analogen Zustand als divergent enthält und somit in die Menge $\{p' \in P' \mid P' \in \text{as-impl}(P) \wedge P'$ kann von p' aus eine unendliche Folge von τs ausführen$\}$ einfügt. □

Die unendliche Folge an τs kann durch eine Schleife an einem durch τs erreichbaren Zustand ausführbar sein oder durch einen Weg, der mit τs ausführbar ist, mit dem unendliche viele Zustände durchlaufen werden. Es ist jedoch zu beachten, dass ein Zustand, von dem aus unendlich viele Zustände durch τs erreichbar sind, nicht divergent sein muss. Es ist auch möglich, dass dieser Zustand eine unendliche Verzweigung hat und somit keine unendlichen Folgen an τs ausführen kann.
Als Erreichbarkeitsbegriff wird wieder die lokale Erreichbarkeit verwendet. Da das Divergieren eines Systems nicht mehr verhindert werden kann, sobald ein divergenter Zustand lokal erreicht werden kann, ist Divergenz als ähnlich „schlimm“ zu bewerten wie ein Fehler-Zustand des Types Kommunikationsfehler.

Definition 5.3 (*fehler-, ruhe- und divergenz-freie Kommunikation*). *Zwei MEIOs P_1 und P_2 kommunizieren* fehler-, ruhe- und divergenz-frei*, wenn keine as-Implementierung ihrer Parallelkomposition P_{12} einen Fehler-, Ruhe- oder Divergenz-Zustand lokal erreichen kann.*

Definition 5.4 (*Divergenz-Verfeinerungs-Basisrelation*). *Für MEIOs P_1 und P_2 mit der gleichen Signatur wird $P_1 \sqsubseteq^{\mathrm{B}}_{Div} P_2$ geschrieben, wenn ein Fehler-, Ruhe- oder Divergenz-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 solcher lokal erreichbar ist. Die* Basisrelation *stell eine* Verfeinerung *bezüglich* Fehler*,* Ruhe *und* Divergenz *dar. $\sqsubseteq^{\mathrm{C}}_{Div}$ bezeichnet die* vollständig abstrakte Präkongruenz *von $\sqsubseteq^{\mathrm{B}}_{Div}$ bezüglich $\cdot\|\cdot$.*

Da nun die grundlegenden Definitionen für Divergenz festgehalten sind, kann man sich einen Begriff für die Traces zu divergenten Zuständen bilden. Da oben bereits festgestellt wurde, dass Divergenz als ähnlich „schlimmer Fehler“ anzusehen ist wie Kommunikationsfehler und dass das Divergieren eines Systems nicht mehr verhinderbar ist, sobald ein divergenter Zustand lokal erreichbar ist, kommt für die Divergenztraces wieder die prune-Funktion zu Einsatz. Ein System, das unendliche viele τs ausführen kann, ist von außen nicht von so einem System zu unterscheiden, das einen Fehler-Zustand der Art Kommunikationsfehler erreicht. Somit wird in den Trace-Mengen auch nicht zwischen Kommunikationsfehler-Traces und Divergenz-Traces explizit unterschieden. Dadurch genügt es nicht mehr nur mit den Kommunikationsfehler-Traces die Sprache fluten, sondern es muss sowohl mit den Kommunikationsfehler-Traces wie auch den Divergenz-Traces geflutet werden. Ebenso werden die strikten Ruhe-Traces mit diesen beiden Trace-Mengen geflutet.

Definition 5.5 (*Divergenz-Traces*). *Sei P ein MEIO und definiere:*

- strikte Divergenz-Traces*:* $StDT(P) := \left\{ w \in \Sigma^* \mid p_0 \stackrel{w}{\Rightarrow}_P p \in Div(P) \right\}$,
- gekürzte Divergenz-Traces*:* $PrDT(P) := \bigcup \{\mathrm{prune}(w) \mid w \in StDT(P)\}$.

Analog zu den Propositionen 3.4 und 4.6 gibt es hier auch eine Proposition, die die Divergenz-Traces eines MEIOs mit den Divergenz-Traces seiner as-Implementierungen verbindet. Die Begründung würde analog wie zu den beiden Propositionen der vorrangige nahten Kapitel laufen, in Kombination mit den Argumenten des Beweises zur Proposition 5.2 in diesem Kapitel.

Proposition 5.6 (*Divergenz-Traces und Implementierung*). *Für ein MEIO P gilt* $StDT(P) = \left\{ w \in \Sigma^* \mid \exists P' \in \text{as-impl}(P) : p'_0 \stackrel{w}{\Longrightarrow}_{P'} p' \in Div(P') \right\}$.

Da die Ruhe-Traces mit den Kommunikationsfehler- und Divergenz-Traces geflutet werden sollen, kann die Ruhe-Semantik nicht aus dem letzten Kapitel übernommen werden auch die geflutete Sprache aus dem Kommunikationsfehler-Kapitel kann nicht übernommen werden. Nur die Kommunikationsfehler-Traces ET können ohne Veränderung auch in diesem Kapitel verwendet werden. Jedoch werden diese Traces im weiteren Verlauf nur innerhalb der größeren Trace-Menge EDT relevant sein.

Definition 5.7 (*Kommunikationsfehler-, Ruhe- und Divergenz-Semantik*). *Sei P ein MEIO.*

- *Die Menge der* Divergenz-Traces *von P ist* $DT(P) := \text{cont}(PrDT(P))$.
- *Die Menge der* Fehler-Divergenz-Traces *von P ist* $EDT(P) := ET(P) \cup DT(P)$.
- *Die Menge der* fehler-divergenz-gefluteten Ruhe-Traces *von P ist* $QDT(P) := StQT(P) \cup EDT(P)$.
- *Die Menge der* fehler-divergenz-gefluteten Sprache *von P ist* $EDL(P) := L(P) \cup EDT(P)$.

Für zwei MEIOs P_1, P_2 mit der gleichen Signatur schreibt man $P_1 \sqsubseteq_{Div} P_2$, wenn $EDT_1 \subseteq EDT_2, QDT_1 \subseteq QDT_2$ und $EDL_1 \subseteq EDL_2$ gilt.

$\sqsubseteq_{Div}$ ist somit keine Einschränkung von $\sqsubseteq_E$ so wie $\sqsubseteq_{Qui}$. Es können Systeme mit einem Kommunikationsfehler nicht von Systemen mit Divergenz unterschieden werden. Da die Basisrelation zwischen diesen Fehler-Arten auch keine Unterscheidung kennt, muss eine sinnvolle Relation dies Eigenschaft auch übernehmen, so wie $\sqsubseteq_{Div}$ dies tut.

Satz 5.8 (*Kommunikationsfehler-, Ruhe- und Divergenz-Semantik für Parallelkompositionen*). *Für zwei komponierbare MEIOs P_1, P_2 und ihre Komposition P_{12} gilt:*

1. $EDT_{12} = \text{cont}(\text{prune}((EDT_1 \| EDL_2) \cup (EDL_1 \| EDT_2)))$,
2. $QDT_{12} = (QDT_1 \| QDT_2) \cup EDT_{12}$,
3. $EDL_{12} = (EDL_1 \| EDL_2) \cup EDT_{12}$.

Beweis. TODO: zu beweisen □

Literaturverzeichnis

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

[BV15] Ferenc Bujtor und Walter Vogler, *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.