

Inhaltsverzeichnis

1	Einleitung	1
2	Definitionen und Notationen	3
2.1	Error-IO-Transitionssystem	3
2.2	Parallelkomposition	4
2.3	Hiding	6
3	Verfeinerung und Error-Freiheit	8
3.1	Präkongruenz für Error	8
3.2	Hiding und Error-Freiheit	17
4	Verfeinerung über Error- und Ruhetraces	19
4.1	Präkongruenz für Ruhe	19
4.2	Hiding für Ruhe	24
4.3	Diskussion für Veränderungen an Semantik oder anderen Definitionen . .	25

1 Einleitung

Der Anfang dieser Arbeit orientiert sich sehr stark an [BV14]. Jedoch wird hier darauf verzichtet die Input-Mengen der Error-IO-Transitionssysteme (EIOs) als disjunkt anzunehmen und alle Definitionen und Sätze werden erst einmal ohne das Verbergen der synchronisierten Aktionen betrachtet.

Dadurch dass die synchronisierten Aktionen nicht verborgen werden, wird hier ein Modell betrachtet, mit dem nicht nur zwei Systeme miteinander kommunizieren können, sondern beliebig viele. Ein Output eines Systems ist somit eine Art Multicast. Jedes System, das diesen Output als Input verarbeiten kann, empfängt ihn somit auch, da bei jeder Komposition der Output weitergeleitet wird an andere Systeme. Kann jedoch ein System den Output nicht als Input aufnehmen, wird dieses System von der Nachricht nicht beeinträchtigt.

Anschließend werden die Auswirkung von Hiding auf diese Struktur untersucht und somit das Verbergen in der Parallelkomposition nachgebildet. Durch das Hiding können Outputs durch interne Aktionen ersetzt werden.

Diese Art der Betrachtung der EIOs wurde auch bereits in [Sch12] gewählt, jedoch wurde diese Arbeit nicht als direkte Quelle genutzt, bis auf den Abschnitt des Hiding. Die Feststellungen im Definitionskapitel und dem Kapitel über Errors stimmen mit dieser Quelle überein, jedoch wurden alle Beweise davon unabhängig neu geführt.

In dieser Arbeit wird ein optimistischer Ansatz für die Erreichbarkeit der Error-Zustände verwendet. Ein Error gilt ist nach der Definition in dieser Arbeit erreichbar, wenn er lokal erreicht werden kann, d.h. durch lokale Aktionen. Die Menge bestehend aus der internen Aktion τ und den Output-Aktionen wird hier als Menge der lokale Aktionen bezeichnet. Alle Elemente aus dieser Menge können ohne weiteres Zutun von außen ausgeführt werden. Somit kann nicht beeinflusst werden, ob diese Transitionen genutzt werden oder nicht. Es besteht also die Möglichkeit, dass das EIO in einen Error-Zustand übergeht, sobald dieser lokal erreichbar ist. Diese Art der Erreichbarkeit von Zuständen wird auch in Kapitel 3 von [BV14] behandelt.

Neben dem hier betrachteten optimistischen Ansatz gibt es noch zwei weitere Ansätze in [BV14] für die Erreichbarkeit von Error-Zuständen: einen hyper-optimistischen Ansatz, bei dem ein Error als erreichbar gilt, wenn er durch interne Aktionen erreicht werden kann, und einen pessimistischen Ansatz, bei dem ein Error als erreichbar gilt, sobald es eine Folge an Inputs und Outputs gibt, mit denen der Error-Zustand vom Startzustand aus erreicht werden kann.

Im Kapitel über die Ruhe-Zuständen wird für die Erreichbarkeit einen hyper-optimistischen Ansatz wählen, da man den Ruhe-Zuständen durch einen Input entrinnen kann und sie somit als nicht so „schlimm“ anzusehen sind wie Errors.

Es wird versucht bei allen betrachteten Zustandsmengen die größte Präkongruenz zu

1 *Einleitung*

finden, die in der jeweiligen Basisrelation enthalten ist und die eine Präkongruenz bezüglich der Parallelkomposition ist.

TODO: erweitern/umformulieren (bis jetzt nur Teile aus anderen Kapitel in Einleitung verschoben)

2 Definitionen und Notationen

Die Definitionen dieses Kapitels sind größtenteils aus [BV14] übernommen, mit den in der Einleitung erwähnten Abänderungen. In diesen Definitionen werden die Grundlagen der Transitionssysteme, mit denen hier gearbeitet werden soll behandelt.

Die nicht mit den entsprechenden Outputs synchronisierten Inputs der zu komponierenden EIOs werden als Inputs der Parallelkomposition übernommen.

2.1 Error-IO-Transitionssystem

Die hier betrachteten EIOs sind Systeme, deren Transitionen mit Inputs und Outputs beschriftet sind. Jede Transition ist dabei mit einem Input oder einem Output versehen. Ebenfalls zulässig ist eine Transitionsbeschriftung mit τ , einer *internen*, unbeobachtbaren *Aktion*. Diese interne Aktion lässt also keine Interaktion mit der Umwelt zu. In [BV14] entsteht das τ in vielen Fällen durch das Verbergen der Inputs und Outputs, die in einer Komposition synchronisiert werden. Hier werden diese Aktionen hingegen nicht verborgen. Jedoch wird im weiteren Verlauf noch das Hiding betrachten, in dem Outputs durch interne Aktionen ersetzt werden.

Definition 2.1 (*Error-IO-Transitionssystem*). Ein Error-IO-Transitionssystem (EIO) ist ein Tupel $S = (Q, I, O, \delta, q_0, E)$, mit den Komponenten:

- Q – die Menge der Zustände,
- I, O – die disjunkten Mengen der (sichtbaren) Input- und Output-Aktionen,
- $\delta \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$ – die Transitionsrelation,
- $q_0 \in Q$ – der Startzustand,
- $E \subseteq Q$ – die Menge der Error-Zustände.

Die Aktionsmenge eines EIOs S ist $\Sigma = I \cup O$ und die Signatur $Sig(S) = (I, O)$.

Um in graphischen Veranschaulichungen Inputs und Outputs zu unterscheiden, wird folgende Notation verwendet: $x?$ für den Input x und $x!$ für den Output x . Falls ein x ohne $?$ oder $!$ verwendet wird, steht dies für eine Aktion, bei der nicht festgelegt ist, ob sie ein Input oder ein Output ist.

Um die Komponenten der entsprechenden Transitionssystem zuzuordnen, werden für die Komponenten die gleichen Indizes wie für ihr zugehöriges System verwendet, z.B. wird I_1 für die Inputmenge des Transitionssystems S_1 geschrieben. Diese Notation wird später analog für die Sprachen eines Systems verwendet.

Die Elemente der Transitionsrelation δ werden wie folgt notieren:

- $p \xrightarrow{\alpha} q$ für $(p, \alpha, q) \in \delta$,
- $p \xrightarrow{\alpha} q$ für $\exists q : (p, \alpha, q) \in \delta$,
- $p \xrightarrow{w} q$ für $p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \dots \xrightarrow{\alpha_n} q$ mit $w \in (\Sigma \cup \{\tau\})^*$, $w = \alpha_1 \alpha_2 \dots \alpha_n$,
- $p \xrightarrow{w} q$ für $p \xrightarrow{\alpha_1 \alpha_2} \dots \xrightarrow{\alpha_n}$ mit $w \in (\Sigma \cup \{\tau\})^*$, $w = \alpha_1 \alpha_2 \dots \alpha_n$,
- $w|_B$ steht für die Zeichenfolge, die aus w entsteht durch Löschen aller Zeichen, die nicht in $B \subseteq \Sigma$ enthalten sind, d.h. es bezeichnet die Projektion von w auf die Menge B ,
- $p \xRightarrow{w} q$ für $w \in \Sigma^*$ mit $\exists w' \in (\Sigma \cup \{\tau\})^* : w'|_{\Sigma} = w \wedge p \xrightarrow{w'} q$,
- $p \xRightarrow{w}$ für $\exists q : p \xRightarrow{w} q$.

Die Sprache von S ist $L(S) = \{w \in \Sigma^* \mid q_0 \xRightarrow{w}\}$.

2.2 Parallelkomposition

Zwei EIOs sind komponierbar, wenn ihre Output-Mengen disjunkt sind. Die Error-Zustände der Parallelkomposition setzen sich aus den Errors der beiden zusammengesetzten Komponenten (geerbte Errors) und den Zuständen, die unsynchronisierbaren Outputs besitzen (neue Errors) zusammen.

In der folgenden Definition muss eine Veränderung gegenüber [BV14] an der Menge der synchronisierten Aktionen vorgenommen werden. Da nicht mehr $I_1 \cap I_2 = \emptyset$ gelten muss, werden die gemeinsamen Inputs synchronisiert.

Definition 2.2 (Parallelkomposition). Zwei EIOs S_1, S_2 sind komponierbar, falls $O_1 \cap O_2 = \emptyset$ gilt. Die Parallelkomposition ist $S_1 \parallel S_2 = (Q, I, O, \delta, q_0, E)$ mit den Komponenten:

- $Q = Q_1 \times Q_2$,
- $I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1)$,
- $O = O_1 \cup O_2$,
- $q_0 = (q_{01}, q_{02})$,
- $\delta = \{((q_1, q_2), \alpha, (p_1, p_2)) \mid (q_1, \alpha, p_1) \in \delta_1, \alpha \in (\Sigma_1 \cup \{\tau\}) \setminus \text{Synch}(S_1, S_2)\} \\ \cup \{((q_1, q_2), \alpha, (q_1, p_2)) \mid (q_2, \alpha, p_2) \in \delta_2, \alpha \in (\Sigma_2 \cup \{\tau\}) \setminus \text{Synch}(S_1, S_2)\} \\ \cup \{((q_1, q_2), \alpha, (p_1, p_2)) \mid (q_1, \alpha, p_1) \in \delta_1, (q_2, \alpha, p_2) \in \delta_2, \alpha \in \text{Synch}(S_1, S_2)\}$,
- $E = (Q_1 \times E_2) \cup (E_1 \times Q_2)$ geerbte Errors
 $\left. \begin{array}{l} \cup \{(q_1, q_2) \mid \exists a \in O_1 \cap I_2 : q_1 \xrightarrow{a} \wedge q_2 \not\xrightarrow{a}\} \\ \cup \{(q_1, q_2) \mid \exists a \in I_1 \cap O_2 : q_1 \not\xrightarrow{a} \wedge q_2 \xrightarrow{a}\} \end{array} \right\}$ neue Errors.

Dabei werden die synchronisierten Aktionen $Synch(S_1, S_2) = (I_1 \cap O_2) \cup (O_1 \cap I_2) \cup (I_1 \cap I_2)$ nicht versteckt, sondern als Outputs bzw. im Fall von $I_1 \cap I_2$ als Inputs der Komposition beibehalten.

S_1 wird einen Partner von S_2 , wenn ihre Parallelkomposition geschlossen ist, d.h. wenn sie duale Signaturen $Sig(S_1) = (I, O)$ und $Sig(S_2) = (O, I)$ haben.

Die Parallelkomposition kann nicht nur für Transitionssysteme betrachtet werden, wie bisher in dieser Arbeit, sondern auch über Aktionsfolgen. *Traces* sind die möglichen Wege des Systems, während ein bestimmtes Wort verarbeitet wird. Dieses Wort besteht aus Inputs und Outputs, mit denen die Folge ab dem Startzustand q_0 beschriftet ist.

Definition 2.3 (Parallelkomposition auf Traces). Gegeben zwei EIOs S_1 und S_2 , sowie $w_1 \in \Sigma_1, w_2 \in \Sigma_2, W_1 \subseteq \Sigma_1^*, W_2 \subseteq \Sigma_2^*$:

- $w_1 \parallel w_2 := \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid w|_{\Sigma_1} = w_1 \wedge w|_{\Sigma_2} = w_2\},$
- $W_1 \parallel W_2 := \bigcup \{w_1 \parallel w_2 \mid w_1 \in W_1 \wedge w_2 \in W_2\}.$

Die Semantik der späteren Kapitel basiert darauf die jeweiligen Zustände, die zu Problemen führen, mit den Traces zu betrachten, mit denen man diese Zustände erreicht. Um dies besser umsetzen zu können, wird eine *prune*-Funktion definiert, die alle Outputs am Ende eines Traces entfernt. Zusätzlich werden Funktionen definiert, die die Traces beliebig fortsetzen.

Definition 2.4 (Pruning- und Fortsetzungsfunktion). Für ein EIO S wird definiert:

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

Für zwei komponierbare EIOs S_1 und S_2 ist ein Ablauf ihrer Parallelkomposition $S_{12} = S_1 \parallel S_2$ eine Transitionsfolge der Form $(p_1, p_2) \xrightarrow{w} (q_1, q_2)$ für ein $w \in \Sigma_{12}^*$. So ein Ablauf kann auf Abläufe von S_1 und S_2 projiziert werden. Diese Projektionen erfüllen $p_i \xrightarrow{w_i} q_i$ mit $w|_{\Sigma_i} = w_i$ für $i = 1, 2$. Umgekehrt sind zwei Abläufe von S_1 und S_2 der Form $p_i \xrightarrow{w_i} q_i$ mit $w|_{\Sigma_i} = w_i$ für $i = 1, 2$, Projektionen von einem Ablauf in S_{12} der Form $(p_1, p_2) \xrightarrow{w} (q_1, q_2)$. Es ist dafür nötig, dass die Abläufe der beiden Systeme und die Systeme selbst komponierbar sind. Das wird dadurch gewährleistet, dass w so gewählt wurde, dass die Projektion auf die einzelnen Alphabete die jeweiligen Wörter ergibt. Falls keine interne Aktionen zugelassen wären, würde sogar nur genau ein Ablauf möglich sein in S_{12} . Da jedoch auch interne Aktionen zulässig sind, sind mehrere Abläufe möglich, da nicht klar ist, wann ein τ in dem Trace ausgeführt wird. Daraus ergibt sich das folgende Lemma.

Lemma 2.5 (Sprache der Parallelkomposition). Für zwei komponierbare EIOs S_1 und S_2 gilt:

$$L_{12} := L(S_1 \parallel S_2) = L(S_1) \parallel L(S_2).$$

2.3 Hiding

Hiding wurde in dem hier verwendeten Kontext bereits in [CJK13] auf Traces betrachtet. Da hier die Betrachtungsweise von Transitionssystemen aus startet, wird auch Hiding aus der Sicht dieser Systeme definieren, wie in [Sch12]. Eine ähnliche Betrachtung für Hiding bei LTS mit Inputs und Outputs wurde auch bereits in [Lyn96] umgesetzt. Dort werden nur Output-Aktionen internalisiert, jedoch gibt es eine Menge an internen Aktionen und nicht nur eine. Das Hiding wird durch einen Internalisierungsoperator umgesetzt. Es sollen dadurch Aktionen versteckt werden können, d.h. durch τ s ersetzt werden. In [CJK13] ist es in der Definition des Hiding möglich Outputs und Inputs zu verstecken. Durch das Verstecken von Outputs sind diese nach außen nicht mehr sichtbar. Werden jedoch Inputs versteckt sind alle Traces, die diesen Input benötigen, nicht mehr ausführbar. Sie sind dann ab dem versteckten Input nicht mehr weiterführbar. Es handelt sich also um echte Einschränkungen des Systems. Die Transitionen werden durch das Hiding von Inputs ähnlich wie bei der Anwendung von Restriktionen in CSS verboten. Diese Art der Einschränkung der Transitionssysteme sollen hier jedoch nicht behandelt werden. Somit wird in der folgenden Definition nur die Internalisierung von Outputs erlaubt, entsprechend Quelle [Sch12].

Definition 2.6 (*Internalisierungsoperator*). Für ein EIO $S = (Q, I, O, \delta, q_0, E)$ ist S/X , mit dem Internalisierungsoperator \cdot/\cdot , definiert als $S' = (Q, I, O', \delta', q_0, E)$ mit:

- $\tau \notin X$.
- $A \subseteq O$
- $O' = O \setminus X$,
- $\delta' = (\delta \cup \{(q, \tau, q') \mid (q, x, q') \in \delta, x \in X\}) \setminus \{(q, x, q') \mid x \in X \cap O\}$,

Die Menge hinter dem Internalisierungsoperator ist in dieser Definition auf Outputs beschränkt. Diese Einschränkung wurde vorgenommen und die weitere Betrachtung zu erleichtern. Jedoch kann es sinnvoll sein die Möglichkeit zu haben dort weitere Aktionen aufnehmen zu können. Dies wird jedoch nicht mehr Teil dieser Arbeit sein.

In [BV14] wird die Parallelkomposition nur mit Verbergen der synchronisierten Aktionen betrachtet, die durch die Synchronisation von einem Input mit einem Output entstehen. Diese Parallelkomposition kann nun mit dem Internalisierungsoperator durch Hiding der synchronisierten Aktionen nachbildet werden, die in der Parallelkomposition zu Outputs werden. Da in dieser Arbeit die Inputmengen der Systeme, die komponiert werden, nicht disjunkt sein müssen, ergeben sich auch Inputs aus der Synchronisation von Aktionen, diese können jedoch mit unserer Definition des Internalisierungsoperators nicht verbor-gen werden. Dies wäre auch nicht sinnvoll, da diese Synchronisation von Inputs keine Kommunikation zwischen den Systemen war, sondern nur eine Zusammenfügung, damit die Parallelkomposition über diesen Input mit weiteren Systemen kommunizieren kann.

Definition 2.7 (*Parallelkomposition mit Internalisierung*). Seien S_1 und S_2 komponierbare EIOs, dann ist $S_1|S_2 = (S_1||S_2)/(Synch(S_1, S_2) \cap O_{12})$.

3 Verfeinerung und Error-Freiheit

3.1 Präkongruenz für Error

Da es in dieser Arbeit vor allem um die Erreichbarkeit und die Kommunikation zwischen EIOs geht, wurden die nächsten beiden Definitionen explizit getrennt und erweitert im Vergleich zu denen in [BV14]. Ebenfalls wurde die Parallelkomposition geändert, wie in [Sch12].

Definition 3.1 (error-freie Kommunikation). *Ein Error ist lokal erreichbar in einem EIO S , wenn $\exists w \in O^* : q_0 \xRightarrow{w} q \in E$.*

Zwei EIOs S_1 und S_2 kommunizieren error-frei, wenn in ihrer Parallelkomposition $S_1 \parallel S_2$ keine Errors lokal erreicht werden können.

Mittels der lokalen Erreichbarkeit von Errors kann ein Verfeinerungsrelation definiert werden. Zusätzlich wird bereits die größte Präkongruenz definiert, die gesucht werden soll.

Definition 3.2 (Error-Verfeinerungs-Basisrelation). *Für EIOs S_1 und S_2 mit der gleichen Signatur wird $S_1 \sqsubseteq_E^B S_2$ geschrieben, wenn ein Error in S_1 nur dann lokal erreichbar ist, wenn er auch in S_2 lokal erreichbar ist. Es handelt sich dabei um eine Basisrelation für die Verfeinerung im Bezug auf Errors.*

\sqsubseteq_E^C bezeichnet die vollständig abstrakte Präkongruenz von \sqsubseteq_E^B bezüglich $\cdot\|\cdot$, d.h. die größte Präkongruenz bezüglich $\cdot\|\cdot$, die in \sqsubseteq_E^B enthalten ist.

Um sich näher mit den Präkongruenzen auseinandersetzen zu können, müssen bestimmte Traces aus der Struktur hervor gehoben werden. Die strikten Errortraces entsprechen Wegen, die direkt vom Startzustand zu einem Zustand in der Menge E führen. Da Outputs Aktionen sind, die von außen nicht verhindert werden können, wird auch noch die Menge der Traces benötigt, die zu einem Zustand führen können, von dem aus mit lokalen Aktionen ein Error erreicht werden kann. Zusätzlich ist auch noch die Menge der Traces interessant, für die es einen Input $a \in I$ gibt, durch den sie möglicherweise nicht fortgesetzt werden können. Diese führen zwar nicht direkt zu einem Error, jedoch in Komposition mit einem anderen Transitionssystem sind dies gefährdete Stellen. Sie führen zu einem neuen Error, sobald dieser Input für die Synchronisation fehlt.

Definition 3.3 (Errortraces). *Für ein EIO S wird definiert:*

- strikte Errortraces: $StET(S) = \{w \in \Sigma^* \mid q_0 \xRightarrow{w} q \in E\}$,
- gekürzte Errortraces: $PrET(S) = \{prune(w) \mid w \in StET(S)\}$,
- Input-kritische Traces: $MIT(S) = \{wa \in \Sigma^* \mid q_0 \xRightarrow{w} q \wedge a \in I \wedge q \not\xrightarrow{a}\}$.

In der folgenden Definition wird festgehalten, was als Errortraces aufgefasst wird. Diese Menge ist dadurch, dass sie die fortgesetzten Traces aus $PrET$ enthält, deutlich allgemeiner als die Menge $StET$. Zusätzlich wird auch noch die geflutete Sprache definiert, in der die Informationen aus der Sprache und den Errortraces vereint werden und somit bei der Inklusion nicht mehr explizit unterscheiden werden.

Definition 3.4 (*Error-Semantik*). Sei S ein EIO.

- Die Menge der Errortraces von S ist $ET(S) := cont(PrET(S)) \cup cont(MIT(S))$.
- Die error-geflutete Sprache von S ist $EL(S) := L(S) \cup ET(S)$.

Für zwei EIOs S_1, S_2 mit der gleichen Signatur wird $S_1 \sqsubseteq_E S_2$ geschrieben, wenn $ET(S_1) \subseteq ET(S_2)$ und $EL(S_1) \subseteq EL(S_2)$ gilt.

Der folgende Satz wurde in [BV14] nur für die Parallelkomposition mit verborgenen synchronisierten Aktionen formuliert, jedoch entspricht er dem analogen Satz aus [Sch12]. Da der Beweis jedoch ohne Beachtung von [Sch12] neu geführt wurde, wird hier eher auf die Erwähnung der Unterschiede zu [BV14] Wert gelegt.

Satz 3.5 (*Error-Semantik für Parallelkompositionen*). Für zwei komponierbare EIOs S_1, S_2 und ihre Komposition $S_{12} = S_1 \parallel S_2$, gilt:

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

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. Dieses Element ist aufgrund der Definition der Menge der Errortraces entweder 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 $(q_{01}, q_{02}) \xrightarrow{x} (q_1, q_2) \wedge a \in I_{12} \wedge (q_1, q_2) \not\xrightarrow{a}$. Da $I_{12} \stackrel{2.2}{=} (I_1 \setminus O_2) \cup (I_2 \setminus O_1) = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$ ist $a \in (I_1 \cup I_2)$ und $a \notin (O_1 \cup O_2)$. Es wird unterscheiden, ob $a \in (I_1 \cap I_2)$ oder $a \in (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ ist. Diese Unterscheidung ist in [BV14] nicht nötig, da dort $I_1 \cap I_2 = \emptyset$ gilt, somit gibt es dort nur den Fall 1b).
 - Fall 1a) ($a \in (I_1 \cap I_2)$): Nun kann den Ablauf der Komposition auf die Transitionssysteme projiziert werden und man erhält dann oBdA $q_{01} \xrightarrow{x_1} q_1 \not\xrightarrow{a}$ und $q_{02} \xrightarrow{x_2} q_2 \not\xrightarrow{a}$ oder $q_{02} \xrightarrow{x_2} q_2 \xrightarrow{a}$ mit $x \in x_1 \parallel x_2$. Daraus kann $x_1 a \in cont(MIT_1) \subseteq ET_1 \subseteq EL_1$ und $x_2 a \in EL_2$ ($x_2 a \in MIT_2$ oder $x_2 a \in L_2$) gefolgert werden. Damit folgt $w \in (x_1 \parallel x_2) \cdot \{a\} \subseteq (x_1 a) \parallel (x_2 a) \subseteq ET_1 \parallel EL_2$, und somit ist w in der rechten Seite der Gleichung enthalten.

- Fall 1b) ($a \in (I_1 \cup I_2) \setminus (I_1 \cap I_2)$): OBdA gilt $a \in I_1$. Durch Projektion erhält man: $q_{01} \xrightarrow{x_1} q_1 \not\xrightarrow{a}$ und $q_{02} \xrightarrow{x_2} q_2$ 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 ($w \in PrET_{12}$): Durch die Definitionen von $PrET$ und $prune$ weiß man, dass es ein $v \in O_{12}^*$ gibt, so dass $(q_{01}, q_{02}) \xrightarrow{w} (q_1, q_2) \xrightarrow{v} (q'_1, q'_2)$ gilt mit $(q'_1, q'_2) \in E_{12}$ und $w = \text{prune}(wv)$. Durch Projektion erhält man $q_{01} \xrightarrow{w_1} q_1 \xrightarrow{v_1} q'_1$ und $q_{02} \xrightarrow{w_2} q_2 \xrightarrow{v_2} q'_2$ mit $w \in w_1 \| w_2$ und $v \in v_1 \| v_2$. Aus $(q'_1, q'_2) \in E_{12}$ folgt, dass es sich entweder um einen geerbten oder einen neuen Error handelt. Bei einem geerbten wäre bereits einer der beiden Zustände q_1 bzw. q_2 ein Error-Zustand gewesen. Der neue Error hingegen wäre durch die fehlende Möglichkeit entstanden, eine synchronisierte Aktion auszuführen.
 - Fall 2a) (geerbter Error): OBdA $q'_1 \in E_1$. Daraus folgt $w_1 v_1 \in StET_1 \subseteq \text{cont}(PrET_1) \subseteq ET_1$. Da gilt $q_{02} \xrightarrow{w_2 v_2}$, 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 Error): OBdA $a \in I_1 \cap O_2$ mit $q'_1 \not\xrightarrow{a} \wedge q'_2 \xrightarrow{a}$. 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 $a \in O_2 \subseteq O_{12}$ 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 $xy \in w_1 \| w_2$. In dem Punkt, dass das präfix-minimale Element noch mit Outputs fortgesetzt werden kann, unterscheidet sich dieser Beweis von dem in [Sch12]. In dieser Quelle wird nicht weiter darauf eingegangen, dass die prune -Funktion an dieser Stelle noch zur Anwendung kommt. Da jedoch später nur Präfixe von x betrachtet werden, ist dieser Unterschied irrelevant. Im Folgenden wird für alle Fälle von xy gezeigt, dass es ein $v \in PrET(S_1 \| S_2) \cup MIT(S_1 \| S_2)$ gibt, das ein Präfix von xy ist und v entweder auf einen Input aus I_{12} endet oder $v = \varepsilon$. Da v entweder leer ist oder auf einen Input endet, 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 hat x ein Präfix in $PrET(S_1 \| S_2) \cup MIT(S_1 \| S_2)$, damit ist x in der Fortsetzung dieser Menge enthalten und somit gilt $x \in ET_{12}$.

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 wird nun davon ausgegangen, 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 S_1 ein Error 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 $v_2 \in MIT_2$ kann durch die gleiche Argumentation ebenfalls geschlossen, dass v'_2 ein Präfix von v_2 ist. Zusätzlich weiß man, dass v_2 auf $b \in I_2$ endet, jedoch muss nicht mehr wie in [BV14] $b \neq a$ gelten. Es kann also keine Aussage mehr darüber getroffen, ob es sich um ein echtes Präfix handelt.

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 . Da nicht mehr $b \neq a$ gelten muss, kann nicht mehr für alle Fälle $q_{02} \xrightarrow{v'_2}$ gefolgert werden, wie das in [BV14] möglich war, sondern nur wenn $a \notin I_2$ gilt.

- Fall I ($v_1 \in MIT_1$ und $v_1 \neq \varepsilon$): Es gibt $q_{01} \xrightarrow{v'_1} q_1 \xrightarrow{a}$ und sei $v' = v''a$. Bei der folgenden Fallunterscheidung müssen bezüglich [BV14] zwei weitere Fälle (Ib) und (Ic)) eingefügt werden, da es zulässig ist, dass a sowohl in I_1 wie auch in I_2 enthalten ist.
 - Fall Ia) ($a \notin \Sigma_2$): Es gilt $q_{02} \xrightarrow{v'_2} q_2$ mit $v'' \in v'_1 \| v'_2$. Dadurch erhält man $(q_{01}, q_{02}) \xrightarrow{v''} (q_1, q_2) \xrightarrow{a}$ 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 $q_{02} \xrightarrow{v''_2} q_2 \xrightarrow{a}$ und $v'' \in v'_1 \| v''_2$. a ist für S_2 , ebenso wie für S_1 , ein fehlender Input. Daraus folgt, dass $(q_1, q_2) \xrightarrow{a}$ gilt. Es wird $v := v''a = v' \in MIT_{12}$ gewählt.
 - Fall Ic) ($a \in I_2$ und $v'_2 \in L_2$): Es gilt $q_{02} \xrightarrow{v''_2} q_2 \xrightarrow{a}$ mit $v'_2 = v''_2 a$. Da jedoch a in $Synch(S_1, S_2)$ liegt, wurde die Menge der synchronisierten Aktionen bezüglich [BV14] erweitert, somit reicht schon, dass $q_1 \xrightarrow{a}$ gilt, um Folgendes schließen zu können $(q_1, q_2) \xrightarrow{a}$. Somit kann hier $v := v''a = v' \in MIT_{12}$ gewählt werden.
 - Fall Id) ($a \in O_2$): Es gilt $v'_2 = v''_2 a$ und $q_{02} \xrightarrow{v''_2}$. Man erhält also $q_{02} \xrightarrow{v''_2} q_2 \xrightarrow{a}$ mit $v'' \in v'_1 \| v''_2$. Daraus ergibt sich $(q_{01}, q_{02}) \xrightarrow{v''} (q_1, q_2)$ mit $q_1 \xrightarrow{a}, a \in I_1, q_2 \xrightarrow{a}, a \in O_2$, somit gilt $(q_1, q_2) \in E_{12}$. Es wird $v := prune(v'') \in PrET_{12}$ gewählt.

- Fall II ($v_1 \in PrET_1$): $\exists u_1 \in O_1^* : q_{01} \xrightarrow{v_1} q_1 \xrightarrow{u_1} q'_1$ mit $q'_1 \in E_1$. Da es hier keine disjunkten Inputmengen wie in [BV14] gibt kann das a , auf das v_1 im Fall $v_1 \neq \varepsilon$ endet, ebenfalls der letzte Buchstabe von v_2 sein. Im Fall von $v_2 \in MIT_2$ kann somit $a = b$ gelten und somit wäre $v_2 = v'_2$. Dieser Fall verläuft jedoch analog zu Fall Ic) und wird somit hier nicht weiter betrachtet. Es gilt somit für alle anderen Fälle hier $q_{02} \xrightarrow{v'_2} q_2$ mit $(q_{01}, q_{02}) \xrightarrow{v'} (q_1, q_2)$.
 - Fall IIa) ($u_2 \in (O_1 \cap I_2)^*, c \in (O_1 \cap I_2)$, sodass u_2c Präfix von $u_1|_{I_2}$ mit $q_2 \xrightarrow{u_2} q'_2 \not\xrightarrow{c}$): Für das Präfix u'_1c von u_1 mit $u'_1c|_{I_2} = u_2c$ weiß man, dass $q_1 \xrightarrow{u'_1} q'_1 \xrightarrow{c}$. Somit gilt $u'_1 \in u'_1|u_2$ und $(q_1, q_2) \xrightarrow{u'_1} (q'_1, q'_2) \in E_{12}$, da für S_2 der entsprechende Input fehlt, der mit dem c Output von S_1 zu koppeln wäre. Es handelt sich also um einen neuen Error. Es wird $v := \text{prune}(v'u'_1) \in PrET(S_1||S_2)$ gewählt, dies ist ein Präfix von v' , da $u_1 \in O_1^*$.
 - Fall IIb) ($q_2 \xrightarrow{u_2} q'_2$ mit $u_2 = u_1|_{I_2}$): Somit ist $u_1 \in u_1|u_2$ und $(q_1, q_2) \xrightarrow{u_1} (q'_1, q'_2) \in E_{12}$, da $q'_1 \in E_1$ und somit handelt es sich um einen geerbten Error. Es wird nun $v := \text{prune}(v'u_1) \in PrET(S_1||S_2)$ gewählt, das wiederum ein Präfix von v' ist.

2.:

Der Beweis für diesen Punkt konnte bezüglich [BV14] fast unverändert übernommen werden, bis auf die Ersetzung der Zeichen der Parallelkomposition.

Es ist durch die Definition klar, dass gilt $L_i \subseteq EL_i$ und $ET_i \subseteq EL_i$. Die Argumentation wird von der rechten Seite der Gleichung aus begonnen:

$$\begin{aligned}
 & (EL_1||EL_2) \cup ET_{12} \\
 & \stackrel{3.4}{=} (L_1 \cup ET_1) || (L_2 \cup ET_2) \cup ET_{12} \\
 & = \underbrace{(L_1||ET_2)}_{\substack{\subseteq (EL_1||ET_2) \\ \stackrel{1.}{\subseteq} ET_{12}}} \cup \underbrace{(ET_1||L_2)}_{\substack{\subseteq (ET_1||EL_2) \\ \stackrel{1.}{\subseteq} ET_{12}}} \cup (L_1||L_2) \cup \underbrace{(ET_1||ET_2)}_{\substack{\subseteq (EL_1||ET_2) \\ \stackrel{1.}{\subseteq} ET_{12}}} \cup ET_{12} \\
 & = (L_1||L_2) \cup ET_{12} \\
 & \stackrel{2.5}{=} L_{12} \cup ET_{12} \\
 & \stackrel{3.4}{=} EL_{12}
 \end{aligned}$$

□

Die folgende Proposition wurde hier noch explizit mit Beweis eingefügt im Gegensatz zu den Ausführungen in [BV14], in denen diese Präkongruenz nur als Folgerung aus dem letzten Satz erwähnt wird. Die Feststellung, dass es sich um eine Präkongruenz handelt, ist wichtig, da dann die erste Eigenschaft erfüllt ist, um eine operationale Beschreibung der vollständig abstrakten Präkongruenz \sqsubseteq_E^C zu erhalten.

Proposition 3.6 (Präkongruenz). \sqsubseteq_E ist eine Präkongruenz bezüglich $\cdot||\cdot$.

Beweis. Es muss gezeigt werden: Wenn $S_1 \sqsubseteq_E S_2$ gilt, dann für jedes komponierbare S_3 auch $S_3 \parallel S_1 \sqsubseteq_E S_3 \parallel S_2$. D.h. es ist zu zeigen, dass aus $ET_1 \subseteq ET_2$ und $EL_1 \subseteq EL_2$ folgt, $ET(S_3 \parallel S_1) \subseteq ET(S_3 \parallel S_2)$ und $EL(S_3 \parallel S_1) \subseteq EL(S_3 \parallel S_2)$. Dies ergibt sich aus der Monotonie von *cont*, *prune* und $\cdot \parallel \cdot$ auf Sprachen wie folgt:

$$\begin{aligned}
 & \bullet \quad ET(S_3 \parallel S_1) \stackrel{3.5 \ 1.}{=} cont(prune((ET_3 \parallel EL_1) \cup (EL_3 \parallel ET_1))) \\
 & \quad \quad \quad \begin{array}{c} ET_1 \subseteq ET_2 \\ \text{und} \\ EL_1 \subseteq EL_2 \\ \subseteq \end{array} cont(prune((ET_3 \parallel EL_2) \cup (EL_3 \parallel ET_2))) \\
 & \quad \quad \quad \stackrel{3.5 \ 1.}{=} ET(S_3 \parallel S_2) \\
 & \bullet \quad EL(S_3 \parallel S_1) \stackrel{3.5 \ 2.}{=} (EL_3 \parallel EL_1) \cup E_{31} \\
 & \quad \quad \quad \begin{array}{c} EL_1 \subseteq EL_2 \\ \text{und} \\ ET_{31} \subseteq ET_{32} \\ \subseteq \end{array} (EL_3 \parallel EL_2) \cup ET_{32} \\
 & \quad \quad \quad \stackrel{3.5 \ 2.}{=} EL(S_3 \parallel S_2)
 \end{aligned}$$

□

In [BV14] wurde auch die Verfeinerung von EIOs als Relation betrachtet mit Spezifikation und Implementierung. Hier soll ebenfalls eine Verfeinerungsrelation über EIOs betrachtet werden, jedoch sollen die synchronisierten Aktionen nicht verborgen werden. Dadurch ändern sich natürlich auch Teile des Beweises, vor allem muss statt mit *StET* mit der Menge *PrET* argumentiert werden. Dieses Lemma existiert in dieser Form nicht in [Sch12], da es dort mit der Aussage von Satz 3.8 kombiniert wurde. Jedoch ist die Aussage dieses Lemmas trotzdem Teil dessen, was gezeigt wird und somit finden sich die Teile dieses Beweises auch dort wieder.

Lemma 3.7 (Verfeinerung mit Errors). *Gegeben sind zwei EIOs S_1 und S_2 mit der gleichen Signatur. Wenn alle Partner EIOs U , die mit S_2 gut kommunizieren, auch mit S_1 gut kommunizieren, dann verfeinert S_1 das EIO S_2 . Diese Verfeinerung entspricht der Relation \sqsubseteq_E von oben: Wenn $U \parallel S_1 \sqsubseteq_E^B U \parallel S_2$ für alle Partner U , dann gilt $S_1 \sqsubseteq_E S_2$.*

Beweis. Da S_1 und S_2 die gleichen Signaturen haben wird $I := I_1 = I_2$ und $O := O_1 = O_2$ definiert. Für jeden der Partner U gilt $I_U = O$ und $O_U = I$.

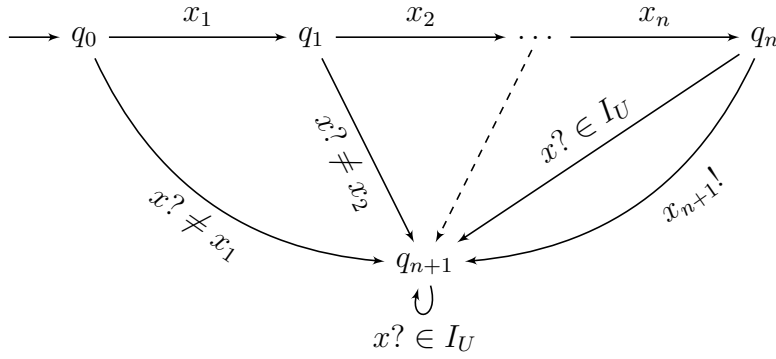
Um $S_1 \sqsubseteq_E S_2$ zu zeigen, wird nachgeprüft, ob folgendes gilt:

- $ET(S_1) \subseteq ET(S_2)$,
- $EL(S_1) \subseteq EL(S_2)$.

Für ein gewähltes präfix-minimales Element $w \in ET(S_1)$ wird gezeigt, dass dieses w oder eines seiner Präfixe in $ET(S_2)$ enthalten ist. Dies ist möglich, da beide Mengen durch *cont* abgeschlossen sind.

- Fall 1 ($w = \varepsilon$): Es handelt sich um einen lokal erreichbaren Error in S_1 . Für U wird ein Transitionssystem verwendet, das nur aus dem Startzustand und einer Schleife für alle Inputs $x \in I_U$ besteht. Somit kann S_1 die gleichen Error-Zustände lokal erreichen wie $U \parallel S_1$. Daraus folgt, dass auch $U \parallel S_2$ einen lokal erreichbaren Error-Zustand haben muss. Durch die Definition von U kann dieser Error nur von S_2 geerbt sein. Es muss also in S_2 ein Error-Zustand durch interne Aktionen und Outputs erreichbar sein, d.h. es gilt $\varepsilon \in PrET(S_2)$.
- Fall 2 ($w = x_1 \dots x_n x_{n+1} \in \Sigma^+$ mit $n \geq 0$ und $x_{n+1} \in I$): Es wird der folgenden Partner U bedachtet (siehe auch Abbildung 3.1):

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


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

Für w können zwei Fälle unterscheiden werden. Beide führen zu $\varepsilon \in PrET(U \parallel S_1)$. Dieses Resultat unterscheidet sich von dem in [BV14], da hier die synchronisierten Aktionen als Outputs vorhanden bleiben und somit kann nicht $\varepsilon \in StET(U \parallel S_1)$ gelten.

- Fall 2a) ($w \in MIT(S_1)$): In $U \parallel S_1$ erhält man $(q_0, q_{01}) \xrightarrow{x_1 \dots x_n} (q_n, q')$ mit $q' \not\stackrel{x_{n+1}}{\rightarrow}$ und $q_n \xrightarrow{x_{n+1}}$. Deshalb gilt $(q_n, q') \in E_{U \parallel S_1}$ und $x_1 \dots x_n \in StET(U \parallel S_1)$. Da alle Aktionen aus w bis auf x_{n+1} synchronisiert werden gilt $x_1, \dots, x_n \in O_{U \parallel S_1}$. Daraus ergibt sich dann $\varepsilon \in PrET(U \parallel S_1)$.
- Fall 2b) ($w \in PrET(S_1)$): In $U \parallel S_1$ erhält man $(q_0, q_{01}) \xrightarrow{u} (q_{n+1}, q'') \xrightarrow{u} (q_{n+1}, q')$ für $u \in O^*$ und $q' \in E_1$. Daraus folgt $(q_{n+1}, q') \in E_{U \parallel S_1}$ und somit $wu \in StET(U \parallel S_1)$. Da alle Aktionen aus w synchronisiert werden, gilt

$x_1, \dots, x_n, x_{n+1} \in O_{U\|S_1}$ und, da $u \in O^*$, folgt $u \in O_{U\|S_1}^*$. Somit ergibt sich $\varepsilon \in PrET(U\|S_1)$.

Da $\varepsilon \in PrET(U\|S_1)$ gilt, kann durch $U\|S_1 \subseteq_E^B U\|S_2$ geschlossen werden, dass auch in $U\|S_2$ ein Error lokal erreichbar sein muss.

Dieser Error kann geerbt oder neu sein.

- Fall 2i) (neuer Error): Da jeder Zustand von U alle Inputs $x \in O = I_U$ zulässt, muss ein lokal erreichbarer Error der Form sein, dass ein Output $a \in O_U$ von U möglich ist, der nicht mit einem passenden Input aus S_2 synchronisiert werden kann. Durch die Konstruktion von U sind in q_{n+1} keine Outputs möglich. Ein neuer Error muss also die Form (q_i, q') haben mit $i \leq n, q' \not\stackrel{x_{i+1}}{\rightarrow}$ und $x_{i+1} \in O_U = I$. Durch Projektion erhält man dann $q_{02} \stackrel{x_1 \dots x_i}{\Rightarrow} q' \not\stackrel{x_{i+1}}{\rightarrow}$ und damit gilt $x_1 \dots x_{i+1} \in MIT(S_2) \subseteq ET(S_2)$. Somit ist ein Präfix von w in $ET(S_2)$ enthalten.
- Fall 2ii) (geerbter Error): U hat $x_1 \dots x_i u$ ausgeführt mit $u \in I_U^* = O^*$ und ebenso hat S_2 diesen Weg ausgeführt. Durch dies hat S_2 einen Zustand in E_2 erreicht, da von U keine Errors geerbt werden können. Es gilt dann $prune(x_1 \dots x_i u) = prune(x_1 \dots x_i) \in PrET(S_2) \subseteq ET(S_2)$. Da $x_1 \dots x_i$ ein Präfix von w ist, führt auch in diesem Fall ein Präfix von w zu einem Error.

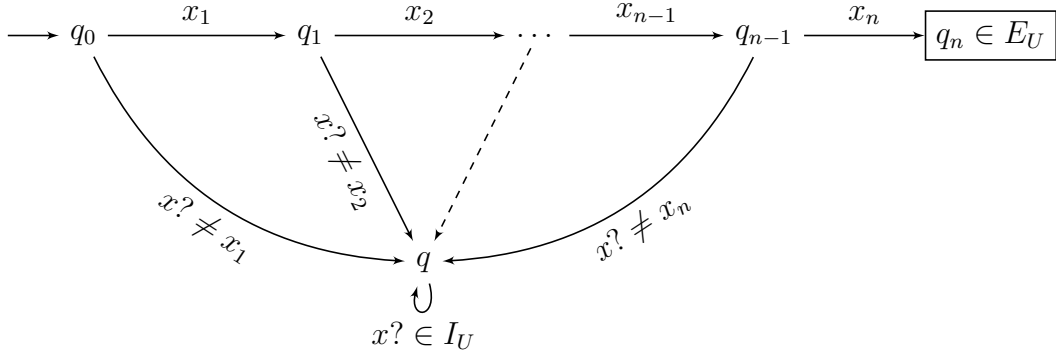
Um die zweiten Inklusion zu beweisen, reicht es aufgrund der ersten Inklusion und der Definition von EL aus zu zeigen, dass $L(S_1) \setminus ET(S_1) \subseteq EL(S_2)$ gilt.

Es wird dafür ein beliebiges $w \in L(S_1) \setminus ET(S_1)$ gewählt und gezeigt, dass es in $EL(S_2)$ enthalten ist.

- Fall 1 ($w = \varepsilon$): Da ε immer in $EL(S_2)$ enthalten ist, ist hier nichts zu zeigen.
- Fall 2 ($w = x_1 \dots x_n$ mit $n \geq 1$): Es wird einen Partner U wie folgt konstruiert (siehe dazu auch Abbildung 3.2):
 - $Q_U = \{q, q_0, q_1, \dots, q_n\}$
 - $q_{0U} = q_0$
 - $E_U = q_n$
 - $\delta_U = \{(q_i, x_{i+1}, q_{i+1}) \mid 0 \leq i < n\}$
 $\cup \{(q_i, x, q) \mid x \in I_U \setminus \{x_{i+1}\}, 0 \leq i \leq n\}$
 $\cup \{(q, x, q) \mid x \in I_U\}$

Da $q_{01} \stackrel{w}{\Rightarrow} q'$ gilt, kann man schließen, dass $U\|S_1$ einen lokal erreichbaren geerbten Error hat. Somit muss $U\|S_2$ ebenfalls einen lokal erreichbaren Error haben.

- Fall 2a) (neuer Error aufgrund von $x_i \in O_U$ und $q_{02} \stackrel{x_1 \dots x_{i-1}}{\Rightarrow} q'' \not\stackrel{x_i}{\rightarrow}$): Es gilt $x_1 \dots x_i \in MIT(S_2)$ und somit $w \in EL(S_2)$. Anzumerken ist, dass nur auf diesem Weg Outputs von U möglich sind, deshalb gibt es keine anderen Outputs von U , die zu einem neuen Error führen können.


 Abbildung 3.2: $x? \neq x_i$ steht für alle $x \in I_U \setminus \{x_i\}$, q_n ist der einzige Error-Zustand

- Fall 2b) (neuer Error aufgrund von $a \in O_2$): Der einzige Zustand, in dem U nicht alle Inputs erlaubt sind, ist q_n , der bereits ein Error-Zustand ist. Falls dieser Zustand erreichbar ist in $U \parallel S_2$, dann besitzt das komponierte EIO einen geerbten Error und es gilt $w \in L(S_2) \subseteq EL(S_2)$, wegen Fall 2c).
- Fall 2c) (geerbter Error von U): Da der einzige Zustand aus E_U q_n ist und alle Aktionen synchronisiert sind, ist dies nur möglich, wenn gilt $q_{02} \xrightarrow{x_1 \dots x_n}$. In diesem Fall gilt, $w \in L(S_2) \subseteq EL(S_2)$.
- Fall 2d) (geerbter Error von S_2): Es gilt dann $q_{02} \xrightarrow{x_1 \dots x_i u} q' \in E_2$ für $i \geq 0$ und $u \in O^*$. Somit ist $x_1 \dots x_i u \in StET(S_2)$ und damit $prune(x_1 \dots x_i u) = prune(x_1 \dots x_i) \in PrET(S_2) \subseteq EL(S_2)$. Somit gilt $w \in EL(S_2)$.

□

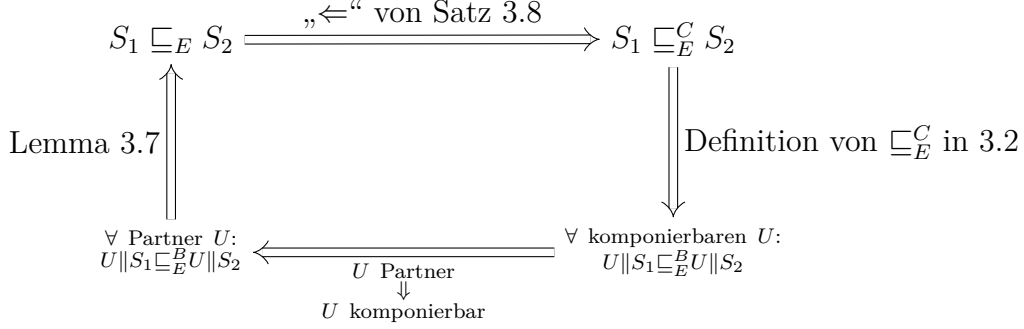
Der folgende Satz sagt aus, dass \sqsubseteq_E die größte Präkongruenz ist, die charakterisiert werden soll, also gleich der vollständig abstrakten Präkongruenz \sqsubseteq_E^C .

Satz 3.8 (Full Abstractness für Error-Semanik). Seien S_1 und S_2 zwei EIOs mit derselben Signatur. Dann gilt $S_1 \sqsubseteq_E^C S_2 \Leftrightarrow S_1 \sqsubseteq_E S_2$, insbesondere ist \sqsubseteq_E eine Präkongruenz.

Beweis. „ \Leftarrow “: Nach Definition gilt, genau dann wenn $\varepsilon \in ET(S)$, ist ein Error lokal erreichbar in S . $S_1 \sqsubseteq_E S_2$ impliziert, dass $\varepsilon \in ET(S_2)$ gilt, wenn $\varepsilon \in ET(S_1)$. Somit ist ein Error in S_1 nur dann lokal erreichbar, wenn dieser auch in S_2 erreichbar ist. Dadurch folgt, dass $S_1 \sqsubseteq_E^B S_2$ gilt, da \sqsubseteq_E^B in Definition 3.2 über die lokale Erreichbarkeit der Error-Zustände definiert wurde. Somit ist \sqsubseteq_E in \sqsubseteq_E^B enthalten. Wie in Proposition 3.6 gezeigt, ist \sqsubseteq_E eine Präkongruenz. Da \sqsubseteq_E^C die größte Präkongruenz bezüglich $\cdot \parallel \cdot$ ist, die in \sqsubseteq_E^B enthalten ist, muss \sqsubseteq_E in \sqsubseteq_E^C enthalten sein. Es folgt also aus $S_1 \sqsubseteq_E S_2$, dass auch $S_1 \sqsubseteq_E^C S_2$ gilt.

„ \Rightarrow “: Durch die Definition von \sqsubseteq_E^C als Präkongruenz in 3.2 folgt aus $S_1 \sqsubseteq_E^C S_2$, dass $U \parallel S_1 \sqsubseteq_E^C U \parallel S_2$ für alle EIOs U , die mit S_1 komponierbar sind. Da \sqsubseteq_E^C nach Definition auch in \sqsubseteq_E^B enthalten sein soll folgt aus $U \parallel S_1 \sqsubseteq_E^C U \parallel S_2$ auch die Gültigkeit von $U \parallel S_1 \sqsubseteq_E^B U \parallel S_2$ für alle diese EIOs U . Mit Lemma 3.7 folgt dann $S_1 \sqsubseteq_E S_2$. □

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



Abbildungung 3.3: Folgerungskette

Aus Satz 3.8 und Lemma 3.7 ergibt sich das folgende Korollar:

Korollar 3.9. *Ein EIO S_1 verfeinert ein EIO S_2 genau dann, wenn für alle EIOs U für die S_2 gut mit U kommuniziert, folgt S_1 kommuniziert ebenfalls gut mit U . Dies lässt sich formal wie folgt ausdrücken: $S_1 \subseteq_E S_2 \Leftrightarrow U \parallel S_1 \subseteq_E^B U \parallel S_2$ für alle Partner U .*

3.2 Hiding und Error-Freiheit

Es soll nun untersucht werden, was für Auswirkungen Hiding auf die Verfeinerungsrelationen hat. Es werden also Outputs der Systeme internalisiert.

Proposition 3.10 (Error-Basisrelation bzgl. Internalisierung). *Wenn $S_1 \subseteq_E^B S_2$ gilt, dann folgt daraus, dass auch $(S_1/X) \subseteq_E^B (S_2/X)$ gilt.*

Beweis. Da die Definition der lokalen Erreichbarkeit auf lokalen Aktionen beruht, die aus den Outputs und der internen Aktion besteht, ändert sich durch das Verbergen von Outputs nichts an der Error-Erreichbarkeit. Somit ist jeder Error, der in S_i lokal erreichbar ist über ein Trace, das Outputs aus X enthält, auch in S_i/X erreichbar, jedoch enthält das Trace nicht mehr diesen Output. Alle Traces, die keine Outputs aus der Menge hinter dem Internalisierungsoperator enthalten, bleiben unverändert erhalten. Es ist auch nicht möglich, dass durch das Verbergen von Outputs neue Errors entstehen. Somit folgt die Behauptung. \square

Satz 3.11 (Präkongruenz bzgl. Internalisierung). *Seien S_1 und S_2 zwei EIOs für die $S_1 \subseteq_E S_2$ gilt, somit gilt auch $(S_1/X) \subseteq_E (S_2/X)$. Daraus folgt insbesondere, dass \subseteq_E eine Präkongruenz bezüglich \cdot/X ist.*

Beweis. Da $S_1 \subseteq_E S_2$ gilt, weiß man, dass $ET_1 \subseteq ET_2$ und $EL_1 \subseteq EL_2$ gilt. Um eine genauere Vorstellung davon zu bekommen, was das Hiding für die Systeme bedeutet,

werden hier zunächst einmal die neuen Errortraces und die neue error-geflutete Sprache definiert. Es gilt:

- $ET(S/X) = \{w \in (\Sigma \setminus (X \cap O))^* \mid \exists w' \in ET(S) : w'|_{\Sigma \setminus (X \cap O)} = w\},$
- $EL(S/X) = \{w \in (\Sigma \setminus (X \cap O))^* \mid \exists w' \in EL(S) : w'|_{\Sigma \setminus (X \cap O)} = w\}.$

Es wird einen präfix-minimaler Traces $w = a_1 a_2 \dots a_m \in ET_1$ gewählt. Daraus folgt, dass ein Ablauf $q_0 \xrightarrow{w} q$ existiert mit $q \in E$ oder q hat nicht für alle Inputs eine Transitionsmöglichkeit. Es gilt also entweder $w \in PrET_1$ oder $w \in MIT_1$. An diesem Punkt muss unterschieden werden, ob es sich hier um ein Trace handelt, der aus dem leeren Wort besteht oder um ein Trace, für den $m \geq 1$ gilt.

- Fall 1 ($w = \varepsilon$): Da Elemente aus MIT_1 mindestens die Länge 1 haben müssen, gilt hier $w \in PrET_1$. Da es in diesem Wort keine Aktionen aus X gibt, werden in diesem Zusammenhang auch keine Aktionen verborgen und somit gilt $\varepsilon \in PrET(S_1/X)$. In $ET(S_1/X)$ sind somit alle beliebigen Kombinationen von Aktionen aus Σ enthalten, genau so wie in ET_1 .
- Fall 2 ($m \geq 1$): Nach der Internalisierung bleiben von dem Ablauf nur noch die Aktionen übrig, die nicht Element von $X \cap O$ sind. Da hier ein präfix-minimales Element aus ET_1 betrachtet wird, muss $a_m \in I$ gelten. Das Trace reduziert sich also auf $w' = w|_{\Sigma \setminus (O \cap X)}$. Dabei bleibt der Input a_m erhalten. w' ist dann in $MIT(S_1/X)$ oder in $PrET(S_1/X)$ enthalten und somit ist w' und alle seine Verlängerungen durch die *cont*-Funktion in $ET(S_1/X)$ enthalten.

Nach Voraussetzung gilt $w \in ET_2$. Nach analoger Argumentation gilt dann auch $w' \in ET(S_2/X)$.

Somit bleibt jetzt nur noch zu zeigen, dass $L(S_1/X) \setminus ET(S_1/X) \subseteq EL(S_2/X)$ gilt. Die Argumentation wieso nur diese Inklusion zu zeigen ist, kann dem Beweis zu Lemma 3.7 entnommen werden. Da bereits bekannt ist, dass $L_1 \setminus ET_1 \subseteq EL_2$ gilt, kann geschlossen werden, dass alle relevanten Traces bereits in EL_2 enthalten sind und die Modifikation durch den Internalisierungsoperator wie oben keine Auswirkungen auf die Teilmengenbeziehung hat. Es ist hier nur noch wichtig festzuhalten, dass für ein Trace aus L_1 keine beliebigen Verlängerungen zulässig sind, genauso wie für $L(S_1/X)$. Jedoch kann immer ein passender Trace in $L(S_1/X)$ zu jedem Trace aus L_1 gefunden werden durch weglassen der Aktionen aus $X \cap O$. \square

Aus 3.6 ist bekannt, dass \sqsubseteq_E eine Präkongruenz bezüglich $\cdot\|\cdot$ ist, und aus 3.11, dass \sqsubseteq_E auch eine Präkongruenz bezüglich \cdot/\cdot ist. Da sich nach Definition 2.7 die Parallelkomposition mit Internalisierung nur aus diesen Operatoren zusammensetzt, erhalten man das folgende Korollar.

Korollar 3.12 (Präkongruenz mit Internalisierung). \sqsubseteq_E ist eine Präkongruenz bezüglich $\cdot|\cdot$.

4 Verfeinerung über Error- und Ruhetraces

4.1 Präkongruenz für Ruhe

In diesem Kapitel wird es nicht mehr nur um die Erreichbarkeit von Error-Zuständen gehen, sondern auch um die Erreichbarkeit von Ruhe-Zuständen. Es wird dabei ein ähnlich vorgehen wie im letzten Kapitel angewandt, jedoch wird [CJK13] als Quelle verwendet. Darin werden ähnliche Konzepte beschrieben, jedoch aus Sicht der Traces. Es werden dort zudem gleichzeitig auch noch Traces mit Divergenz betrachtet. Diese Eigenschaft der Zustände wird hier zunächst nicht betrachtet.

Die Zustände, die keine Outputs und keine Transitionsmöglichkeit für eine interne Aktion haben werden, als eine Art Verklemmung angesehen, da sie ohne einen Input von einem Kommunikationspartner den Zustand nicht mehr verlassen können.

Definition 4.1 (*Ruhe*). Ein Ruhe-Zustand ist ein Zustand in einem EIO, der keine Outputs und kein τ zulässt.

Somit ist die Menge der Ruhe-Zustände in einem EIO wie folgt formal definiert: $Q_{ui} := \{q \in Q \mid \forall \alpha \in (O \cup \{\tau\}) : q \not\stackrel{\alpha}{\rightarrow}\}$.

Für die Erreichbarkeit wird wie im letzten Kapitel wieder der optimistische Ansatz der lokalen Erreichbarkeit für die Error-Zustände verwendet. Ruhe ist kein unabwendbarer Fehler, sondern kann durch einen Input repariert werden. Somit ist ein Ruhe-Zustand als nicht so „schlimmer Fehler“ anzusehen wie ein Error. Somit ist ein Ruhe-Zustand ebenso wie ein Error-Zustand erreichbar, sobald er durch Outputs und τ s erreicht werden kann, jedoch ist nicht jede beliebige Fortsetzung eines Traces, der die Eigenschaft erfüllt ein Ruhe-Trace.

Definition 4.2 (*error- und ruhe-freie Kommunikation*). Zwei EIOs S_1 und S_2 kommunizieren error- und ruhe-frei, wenn in ihrer Parallelkomposition $S_1 \parallel S_2$ keine Errors und keine Ruhe-Zustände lokal erreichbar sind.

Definition 4.3 (*Ruhe-Verfeinerungs-Basisrelation*). Für EIOs S_1 und S_2 mit der gleichen Signatur wird $S_1 \sqsubseteq_{Q_{ui}}^B S_2$ geschrieben, wenn ein Error oder Ruhe-Zustand in S_1 nur dann lokal erreichbar ist, wenn ein solcher auch in S_2 lokal erreichbar ist. Diese Basisrelation stellt eine Verfeinerung bezüglich Errors und Ruhe-Zustände dar.

$\sqsubseteq_{Q_{ui}}^C$ bezeichnet die vollständig abstrakte Präkongruenz von $\sqsubseteq_{Q_{ui}}^B$ bezüglich $\cdot \parallel \cdot$.

Um eine genauer 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 grösste Präkongruenz finden und definieren zu können. Wie bereits oben erwähnt, sind Ruhe-Zustände reparierbar im Gegensatz zu Errors. Somit werden keine gekürzten Ruhetraces benötigt, bei der die *prune*-Funktion zur Anwendung käme, da auch keine beliebigen Verlängerungen davon benötigt werden.

Definition 4.4 (*Ruhetraces*). Sei S ein EIO und definiere:

- strikte Ruhetraces: $StQT(S) := \{w \in \Sigma^* \mid q_0 \xRightarrow{w} q \in Qui\}$.

Nur für die Ruhe wird eine neue Semantik definiert, die Error-Semantik wird aus dem letzten Kapitel übernommen. Somit gelten für ET und EL die Definitionen aus dem letzten Kapitel.

Definition 4.5 (*Ruhe-Semantik*). Sei S ein EIO.

- Die Menge der error-gefluteten Ruhetraces von S ist $QT(S) := StQT(S) \cup ET(S)$.

Für zwei EIOs S_1, S_2 mit der gleichen Signatur wird $S_1 \sqsubseteq_{Qui} S_2$ geschrieben, wenn $S_1 \sqsubseteq_E S_2$ und $QT(S_1) \subseteq QT(S_2)$ gilt.

Für die Menge der error-gefluteten Ruhetraces QT wurde eine Informationsvermischung mit den Errortraces vorgenommen wie beim Fluten der Sprache EL . Da jedoch durch die Ruhetraces keine neuen Traces entstehen, die nicht bereits in der gefluteten Sprache EL enthalten wären, würde eine neue Flutung nichts ändern. Es wird also durch die Relation \sqsubseteq_{Qui} nur die bereits existierende Präkongruenz \sqsubseteq_E eingeschränkt.

Das folgende Lemma soll explizit festhalten, wie Ruhezustände sich unter der Parallelkomposition verhalten. Dies ist vor allem in dem danach folgenden Satz relevant.

Lemma 4.6 (*Ruhe-Zustände unter Parallelkomposition*). 1. Ein Zustand (q_1, q_2) aus der Parallelkomposition $S_{12} = S_1 \parallel S_2$ ist ruhig, wenn es auch die Zustände q_1 und q_2 in S_1 bzw. S_2 sind.

2. Wenn der Zustand (q_1, q_2) ruhig ist und nicht in E_{12} enthalten ist, dann sind auch die auf die Teilsysteme projizierten Zustände q_1 und q_2 ruhig.

Beweis.

1.:

Da $q_1 \in Qui_1$ und $q_2 \in Qui_2$ gilt, haben diese beiden Zustände jeweils höchstens die Möglichkeiten Transitionen auszuführen, die mit Inputs beschriftet sind. Jedoch keine Möglichkeiten für Outputs oder τ s. Der Zustand, der durch Parallelkomposition aus diesen beiden Zuständen entsteht hat die Transitionsmöglichkeiten dieser Zustände die parallel ausgeführt werden. Es gibt somit drei unterschiedliche Möglichkeiten:

- Fall 1 ($a \in I_i \setminus Synch(S_1, S_2) \wedge q_i \xrightarrow{a}$ für ein $i \in \{1, 2\}$): Da a ein unsynchronisierter Input ist, wird dieser unabhängig ausgeführt, somit hat (q_1, q_2) ebenfalls die Möglichkeit eine Transition mit a als Input auszuführen.
- Fall 2 ($a \in I_1 \cap I_2 \wedge q_i \xrightarrow{a}$ für beide $i \in \{1, 2\}$): Da beide Zustände den gleichen Input ausführen können, der zu synchronisieren ist, gilt somit $(q_1, q_2) \xrightarrow{a}$. Wobei $a \in I_{12}$ gilt.

- Fall 3 ($a \in I_i \cap \text{Synch}(S_1, S_2) \wedge q_i \xrightarrow{a}$ für ein $i \in \{1, 2\}$): Hier handelt es sich um eine Transition, die für (q_1, q_2) nicht ausführbar ist, da es keine passende Transition für das andere System gibt, mit dem diese Aktion synchronisiert werden könnte. Da es sich hier jedoch um den Input der zu synchronisierenden Aktion handelt, entsteht daraus auch kein neuer Fehler, sondern einfach nur eine Transition, die nicht genommen werden kann, weil der passende Output nicht vorhanden ist.

Dies sind alle Fälle, die auftreten können für die Parallelkomposition der Transitionsmöglichkeiten. Bei keinen dieser Möglichkeiten ist ein Output oder ein τ entstanden und somit hat der Zustand (q_1, q_2) auch keine Möglichkeiten solche Transitionen auszuführen. Daraus folgt also, dass $(q_1, q_2) \in \text{Qui}_{12}$ gilt.

2.:

Es gilt $(q_1, q_2) \in \text{Qui}_{12} \setminus E_{12}$, somit hat dieser Zustand maximal die Möglichkeit Transitionen für Inputs auszuführen. Diese Transitionen für Inputs können nur aus Inputs, die nicht in der Menge der synchronisierten Handlungen enthalten sind oder aus der Synchronisation von zwei Inputs aus der Menge $I_1 \cup I_2$ entstehen. Durch die Parallelkomposition von Outputs mit anderen Aktionen können keine Inputs entstehen und in der Definition der Parallelkomposition auch keine τ s. Outputs, die nicht in der Menge $\text{Synch}(S_1, S_2)$ enthalten sind, würden als Outputs des zusammengesetzten Zustandes übernommen werden und können somit weder bei q_1 noch bei q_2 vorhanden sein. Falls jedoch $(q_1, q_2) \in E_{12}$ zulassen wäre, wäre es möglich, dass einer der beiden Zustände q_i eine Transitionsmöglichkeit für einen Output hat, der aufgrund eines fehlenden Inputs des anderen Zustandes nicht synchronisiert werden kann und somit ein neuer Error entsteht. Dieser Fall wird jedoch durch die zusätzliche Einschränkung ausgeschlossen. Falls einer der beiden Zustände eine Transitionsmöglichkeit für ein τ gehabt hätte, müsste dies auch der zusammengesetzte Zustand haben und könnte somit nach Definition nicht ruhig sein. Es folgt also, dass q_1 und q_2 ebenso nur Transitionen mit Inputs ausführen können. Es gilt also $q_1 \in \text{Qui}_1$ und $q_2 \in \text{Qui}_2$. \square

In dem folgenden Satz sind Punkt 1. und 3. nur zur Vollständigkeit aufgeführt. Sie entsprechen Punkt 1. und 2. aus Satz 3.5.

Satz 4.7 (Error- und Ruhe-Semantik für Parallelkompositionen). Für zwei komponierbare EIOs S_1, S_2 und ihre Komposition $S_{12} = S_1 \parallel S_2$ gilt:

1. $ET_{12} = \text{cont}(\text{prune}((ET_1 \parallel EL_2) \cup (EL_1 \parallel ET_2)))$,
2. $QT_{12} = (QT_1 \parallel QT_2) \cup ET_{12}$,
3. $EL_{12} = (EL_1 \parallel EL_2) \cup ET_{12}$.

Beweis. Wir beweisen nur den 2. Punkt.

„ \subseteq “:

Hier muss unterscheiden werden ob ein $w \in \text{StQT}_{12} \setminus ET_{12}$ oder ein $w \in ET_{12}$ betrachtet wird. Im zweiten Fall ist das w in der rechten Seite enthalten. Somit wird ab jetzt ein $w \in \text{StQT}_{12} \setminus ET_{12}$ betrachtet und dessen Zugehörigkeit zur rechten Menge versucht

zu zeigen. Aufgrund von Definition 4.4 weiß man, dass $(q_{01}, q_{02}) \xrightarrow{w} (q_1, q_2)$ gilt mit $(q_1, q_2) \in Qui_{12}$. Durch Projektion erhält man $q_{01} \xrightarrow{w_1} q_1$ und $q_{02} \xrightarrow{w_2} q_2$ mit $w \in w_1 \| w_2$. Aus $(q_1, q_2) \in Qui_{12}$ kann mit Punkt 2. von Lemma 4.6 folgert werden, dass bereits $q_1 \in Qui_1$ und $q_2 \in Qui_2$ gilt. Somit gilt $w_1 \in StQT_1 \subseteq QT_1$ und $w_2 \in StQT_2 \subseteq QT_2$. Daraus folgt dann $w \in QT_1 \| QT_2$ und somit ist w in der rechten Seite der Gleichung enthalten.

„ \supseteq “:

Es muss wieder danach unterschieden werden aus welcher Menge das betrachtete Element stammt. Falls $w \in ET_{12}$ gilt, so kann die Zugehörigkeit zur linken Seite direkt folgert werden. Wird für den weiteren Beweis dieser Inklusionsrichtung ein Element $w \in QT_1 \| QT_2$ betrachtet und gezeigt, dass es in der linken Menge enthalten ist. Da $QT_i = StQT_i \cup ET_i$ 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 $w_2 \in StQT_2 \subseteq L_2$ gelten oder $w_2 \in ET_2$ und somit gilt auf jeden Fall $w_2 \in EL_2$. Daraus kann dann mit dem ersten Punkt von Satz 3.5 bzw. aus dem ersten Punkt dieses Satzes gefolgert werden, dass $w \in ET_{12}$ gilt und somit in der linken Seite der Gleichung enthalten ist.
- Fall 1 ($w_1 \in StQT_1 \wedge w_2 \in StQT_2$): Es gilt in diesem Fall $q_{01} \xrightarrow{w_1} q_1 \in Qui_1$ und $q_{02} \xrightarrow{w_2} q_2 \in Qui_2$. Da q_1 und q_2 in der Ruhe-Menge enthalten sind, ist auch der Zustand, der aus ihnen zusammengesetzt ist, in der Parallelkomposition ruhig und lässt keine τ -Transitionen zu, wie bereits in Punkt 1. von Lemma 4.6 gezeigt. Es gilt also für die Komposition $(q_{01}, q_{02}) \xrightarrow{w} (q_1, q_2) \in Qui_{12}$ und dadurch ist w in der linken Seite der Gleichung enthalten, da $w \in StQT_{12} \subseteq QT_{12}$ gilt.

□

Die folgende Proposition ist eine direkte Folgerung aus dem letzten Satz. Jedoch ist es eine wichtige Feststellung für die weiteren Verlauf die größte Präkongruenz finden zu wollen.

Proposition 4.8 (Präkongruenz). \sqsubseteq_{Qui} ist eine Präkongruenz bezüglich $\cdot \| \cdot$.

Beweis. Es muss gezeigt werden: Wenn $S_1 \sqsubseteq_{Qui} S_2$ gilt, so auch $S_3 \| S_1 \sqsubseteq_{Qui} S_3 \| S_2$ für jedes S_3 . D.h. es ist zu zeigen, dass aus $S_1 \sqsubseteq_E S_2$ und $QT_1 \subseteq QT_2$ folgt, dass $S_{31} \sqsubseteq_E S_{32}$ und $QT_{31} \subseteq QT_{32}$. Dies ergibt sich wie im Beweis zu Proposition 3.6 aus der Monotonie von $\cdot \| \cdot$ auf Sprachen wie folgt:

$$\begin{array}{c}
 \text{Proposition 3.6} \\
 \text{und} \\
 S_1 \sqsubseteq_E S_2 \\
 \bullet \quad S_{31} \quad \sqsubseteq_E \quad S_{32}
 \end{array}$$

$$\begin{aligned}
 \bullet \quad QT_{31} &\stackrel{4.7}{=}^2 (QT_3 \parallel QT_1) \cup ET_{31} \\
 &\quad \begin{array}{c} ET_{31} \subseteq ET_{32} \\ \text{und} \\ QT_1 \subseteq QT_2 \\ \subseteq \end{array} (QT_3 \parallel QT_2) \cup ET_{32} \\
 &\stackrel{4.7}{=}^2 QT_{32}
 \end{aligned}$$

□

Lemma 4.9 (Verfeinerung mit Ruhe-Zuständen). *Gegeben sind zwei EIOs S_1 und S_2 mit der gleichen Signatur. Wenn alle Partner EIOs U , die mit S_2 gut kommunizieren, auch mit S_1 gut kommunizieren, dann verfeinert S_1 das EIO S_2 . Diese Verfeinerung entspricht der Relation \sqsubseteq_{Qui} von oben: Wenn $U \parallel S_1 \sqsubseteq_{Qui}^B U \parallel S_2$ für alle Partner U , dann gilt $S_1 \sqsubseteq_{Qui} S_2$.*

Beweis. Da davon ausgehen wird, dass S_1 und S_2 die gleiche Signatur haben, definiert man $I := I_1 = I_2$ und $O := O_1 = O_2$. Für jeden Partner U gilt $I_U = O$ und $O_U = I$. Um zu zeigen, dass die Relation $S_1 \sqsubseteq_{Qui} S_2$ gilt, müssen die folgenden Punkte nachweisen werden:

- $S_1 \sqsubseteq_E S_2$,
- $QT(S_1) \subseteq QT(S_2)$.

Der erste Punkt wurde bereits in Lemma 3.7 gezeigt. So bleibt nur noch die Inklusion $QT_1 \subseteq QT_2$ zu zeigen. Diese Inklusion kann jedoch noch analog zum Beweis der Inklusion der gefluteten Sprachen in Lemma 3.7 weiter einschränken. Da bereits bekannt ist, dass $ET_1 \subseteq ET_2$ gilt, muss nur noch $StQT_1 \subseteq QT_2$ gezeigt werden.

Es wird ein $w \in StQT(S_1)$ gewählt und gezeigt, dass es auch in $QT(S_2)$ enthalten ist. Es ist vom Startzustand von S_1 durch das Wort w ein ruhiger Zustand erreichbar. Dies hat keine Auswirkungen auf die Parallelkomposition $U \parallel S_1$, wenn in U kein Ruhe-Zustand durch w erreicht wird. Somit wird der Partner U betrachtet bei dem durch w ebenfalls ein Ruhe-Zustand erreicht wird. Daraus folgt dann, dass in $U \parallel S_1$ ein Ruhe-Zustand mit w erreicht wird und durch die gegebene Relation folgt dann auch, dass in $U \parallel S_2$ auch ein Ruhe-Zustand durch w erreicht werden muss. Dies kann nur der Fall sein, wenn in S_2 bereits ein Ruhe-Zustand durch w erreicht wird. Somit gilt also $w \in StQT_2$. □

Mit dem folgenden Satz wird festgehalten, dass mit \sqsubseteq_{Qui} die größte Präkongruenz gefunden wurde bezüglich $\cdot \parallel \cdot$ die in \sqsubseteq_{Qui}^B enthalten ist.

Satz 4.10 (Full Abstractness für Ruhe-Semantik). *Seien S_1 und S_2 zwei EIOs mit derselben Signatur. Dann gilt $S_1 \sqsubseteq_{Qui}^C S_2 \Leftrightarrow S_1 \sqsubseteq_{Qui} S_2$, insbesondere ist \sqsubseteq_{Qui} eine Präkongruenz.*

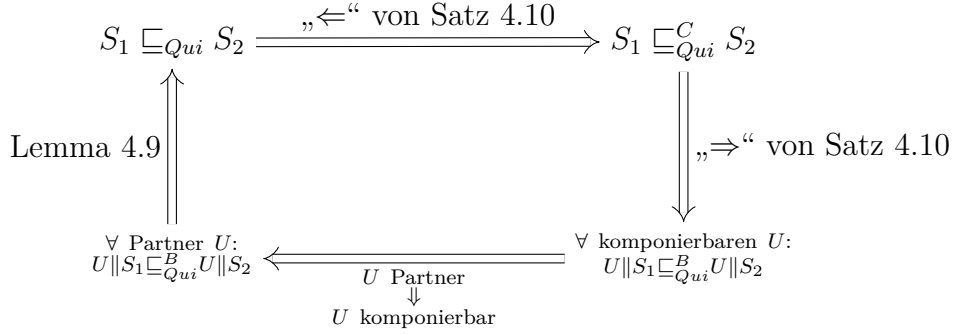
Beweis. Wie bereits in Proposition 4.8 festgehalten, ist \sqsubseteq_{Qui} eine Präkongruenz.

„ \Leftarrow “: Nach Definition gilt, wenn $\varepsilon \in QT(S)$, ist in S ein Ruhe-Zustand durch interne Aktionen oder ein Error-Zustand lokal erreichbar. Somit impliziert $S_1 \sqsubseteq_{Qui} S_2$, dass

$\varepsilon \in QT_2$ gilt, wenn $\varepsilon \in QT_1$. Daraus folgt ebenfalls, dass $S_1 \sqsubseteq_{Qui}^B S_2$ gilt. Somit folgt aus $S_1 \sqsubseteq_{Qui} S_2$ der relationale Zusammenhang $S_1 \sqsubseteq_{Qui}^C S_2$.

„ \Rightarrow “: Durch die Definition von \sqsubseteq_{Qui}^C folgt aus $S_1 \sqsubseteq_{Qui}^C S_2$, dass $U \parallel S_1 \sqsubseteq_{Qui}^C U \parallel S_2$ für alle EIOs U , die mit S_1 komponierbar sind. Somit folgt auch die Gültigkeit von $U \parallel S_1 \sqsubseteq_E^B U \parallel S_2$ für alle diese EIOs U . Mit Lemma 4.9 folgt dann $S_1 \sqsubseteq_{Qui} S_2$. \square

Es wurde somit, wie im letzten Kapitel, eine Kette an Folgerungen gezeigt, die sich zu einem Ring schließen. Dies ist in Abbildung 4.1 dargestellt.



Abbildungung 4.1: Folgerungskette

Aus Satz 4.10 und Lemma 4.9 erhält man das folgende Korollar:

Korollar 4.11. *Ein EIO S_1 verfeinert einen EIO S_2 genau dann, wenn für alle EIOs U für die S_2 gut mit U kommuniziert folgt S_1 kommuniziert ebenfalls gut mit U . Dies lässt sich formal wie folgt ausdrücken: $S_1 \sqsubseteq_{Qui} S_2 \Leftrightarrow U \parallel S_1 \sqsubseteq_{Qui}^B U \parallel S_2$ für alle Partner U .*

4.2 Hiding für Ruhe

Es soll nun auch hier die Auswirkungen der Internalisierung von Aktionen auf die Verfeinerungsrelationen untersucht werden. Es werden Outputs in interne Aktionen umgewandelt. Da jedoch bei den Ruhe-Zuständen auch τ -Transitionen verboten wurden, verändert sich nichts an der Menge der ruhigen Zustände. Da die Erreichbarkeit von Ruhe-Zuständen mittels interner Aktionen betrachtet wurde, können durch das verbergen von Outputs neue erreichbare Ruhe-Zustände hinzu kommen. Somit ist es nicht möglich hier eine analoge Proposition zu 3.10 zu formulieren. Es kann zwar daraus geschlossen werden, dass alle Ruhe-Zustände, die vor dem Hiding zu erreichen waren mit τ s auch danach noch erreichbar sind, jedoch können durch die Internalisierung neue erreicht werden, die möglicherweise in dem anderen Transitionssystem nicht erreicht werden können. Da jedoch für die Präkongruenz \sqsubseteq_{Qui} noch Wissen über die Teilmengenbeziehung der Traces existiert, kann einen Satz analog zu 3.11 formulieren.

Satz 4.12 (Präkongruenz bzgl. Internalisierung). *Seien S_1 und S_2 zwei EIOs für die $S_1 \sqsubseteq_{Qui} S_2$ gilt, somit gilt auch $(S_1/\{x_1, x_x, \dots, x_n\}) \sqsubseteq_{Qui} (S_2/\{x_1, x_x, \dots, x_n\})$. Es ist also \sqsubseteq_{Qui} eine Präkongruenz bezüglich \cdot/\cdot .*

Beweis. Da $S_1 \sqsubseteq_{Qui} S_2$ gilt, kann geschlossen werden, dass $S_1 \sqsubseteq_E S_2$ und $QT_1 \subseteq QT_2$ gilt. Aufgrund von Satz 3.11 ist bekannt, dass daraus $(S_1/\{x_1, x_x, \dots, x_n\}) \sqsubseteq_E (S_2/\{x_1, x_x, \dots, x_n\})$ folgt. Ebenso wie im Beweis zu Satz 3.11 kann hier davon ausgehen werden, dass $X \subseteq O$ gilt. Für jeden Trace aus QT_1 kann ein passender in $QT(S_1/X)$ gefunden werden und ebenso für das Transitionssystem S_2 . Die Argumentation läuft hierfür analog zum Beweis von Satz 3.11. Es ist hier auch kein Problem, dass neue Zustände hinzu kommen, von denen aus durch τ s Ruhe-Zustände erreichbar sind, da diese τ s aus Outputs entstanden sind, die verborgen wurden. Es war also bereits ein Trace in QT_i vorhanden, der nur aus Outputs aus X bestanden hat und der dann durch das Hiding zu einem τ wurde, was aber das passende Trace ist, den man zu dem ursprünglichen in $QT(S_i/X)$ findet. Somit gilt also auch $QT(S_1/X) \subseteq QT(S_2/X)$. Daraus folgt dann, dass die Relation \sqsubseteq_{Qui} trotz Hiding erhalten bleibt und somit das Hiding bezüglich dieser Relation eine Präkongruenz ist. \square

In Definition 2.7 wurde mit Hilfe des Internalisierungsoperator aus der Parallelkomposition ohne Verbergen die Parallelkomposition mit Verbergen der synchronisierten Aktionen nachgebildet. Es kann deren Eigenschaft als Präkongruenz aus den Präkongruenzeigenschaften von $\cdot\|\cdot$ und \cdot/\cdot bezüglich \sqsubseteq_{Qui} aus der Proposition 4.8 und dem Satz 4.12 geschlossen werden.

Korollar 4.13 (Präkongruenz mit Internalisierung). *\sqsubseteq_{Qui} ist eine Präkongruenz bezüglich $\cdot|\cdot$.*

4.3 Diskussion für Veränderungen an Semantik oder anderen Definitionen

Es wäre hier auch denkbar, dass man auch für die Ruhe-Zustände einen lokalen Erreichbarkeitsbegriff verwendet. Es könnte dadurch sogar möglich sein eine noch größere Präkongruenz zu erhalten. Dies wurde hier jedoch nicht gemacht, da es bei den denkbaren Umsetzung eines solchen Erreichbarkeitsbegriffes zu Problemen kommt, so dass nicht mehr alle Inklusionen erreichen können, die in den Sätzen gefordert wurden.

Die Umsetzung des lokalen Erreichbarkeitsbegriffes wie im letzten Kapitel, mit abschneiden der Outputs und beliebigem Fortsetzen der Traces, würde zu einem Problem in Satz 4.7 in Punkt 2. führen, dass analog zu dem Problem wäre in der Semantik, die als nächstes beschrieben wird. Zusätzlich könnte man nicht die EL als geflutete Sprache beibehalten sondern müsste noch zusätzlich mit den Ruhetraces fluten, da sonst nicht mehr alle Informationen, die in Ruhetraces enthalten auch in der gefluteten Sprache enthalten wäre. Durch diese Abänderung käme es zusätzlich noch zu einem Problem in Punkt 3. bei Satz 4.7. Durch die Anwendung der Definition der gefluteten Sprachen

würden Terme entstehen, in denen Ruhetraces und Traces aus der Sprache L in Parallelkomposition gesetzt würden. Da die Ruhetraces nicht mehr Teil der Sprache sind, kann nicht mehr gelten, dass die Teil von $L_1 \parallel L_2$ ist. Es kann auch nicht Teil der Ruhetraces sein, da Ruhetraces nur aus Kombination von zwei Ruhetraces entstehen können oder in dem sie Errortraces sind, was beides hier nicht der Fall ist.

Man muss jedoch nicht an dieser Umsetzung für die lokale Erreichbarkeit festhalten. Man könnte sich auch eine Umsetzung denken, in dem man die Ruhetraces zwar um die Outputs kürzt, jedoch dann nur Fortsetzungen zulässt, die bereits im Transitionssystem möglich sind. Somit benötigt man keine weitere Flutung der gefluteten Sprache EL , da weiterhin alle Ruhetraces in der Sprache L enthalten sind. Jedoch stößt man hier wieder bei Punkt 2. von Satz 4.7 an Probleme. Hierzu muss davon ausgegangen werden, dass eine neue *prune*-Funktion definiert wurde bzw. eine neue *cont*-Funktion, wie z.B. $prune' : \Sigma^* \rightarrow \mathfrak{P}(\Sigma^*), w \mapsto \{u \mid \exists v \in O^* : w = uv\}$. Somit würde sich die Definition der Ruhetraces entsprechend abändern. So das statt *StQT* ein entsprechendes *PrQT* enthalten wäre, dass durch die Anwendung der *prune'*-Funktion entstanden wäre. Somit wäre Punkte 2. von Satz 4.7 entsprechend auch leicht abgewandelt und müsste $QT_{12} = prune'(QT_1 \parallel QT_2) \cup ET_{12}$ lauten. Das Problem würde nun auftauchen wenn man die Inklusionsrichtung \supseteq zu beweisen versucht. Hierzu gibt es jedoch folgendes Gegenbeispiel, dass die Behauptung gar nicht stimmen kann.

Für S_1 wird das folgende Transitionssystem verwendet (siehe dazu auch Abbildung 4.2):

- $Q_1 = \{q_{01}, q_{11}\}$
- $I_1 = \emptyset$
- $O_1 = \{a\}$
- $\delta_1 = \{(q_{01}, a, q_{11})\}$
- $E_1 = \emptyset$
- $Qui_1 = \{q_{11}\}$

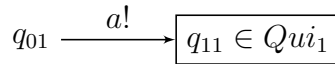
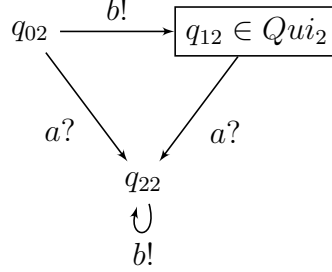


Abbildung 4.2: S_1

Für S_2 wird das folgende Transitionssystem verwendet (siehe dazu auch Abbildung 4.3):

- $Q_2 = \{q_{02}, q_{12}, q_{22}\}$
- $I_2 = \{a\}$
- $O_2 = \{b\}$

- $\delta_2 = \{(q_{02}, a, q_{22}), (q_{02}, b, q_{12}), (q_{12}, a, q_{22}), (q_{22}, b, q_{22})\}$
- $E_2 = \emptyset$
- $Qui_2 = \{q_{12}\}$


 Abbildung 4.3: S_2

Somit ergibt sich für die Parallelkomposition $S_{12} = S_1 \parallel S_2$ das folgende Transitionssystem, bei dem bereits unerreichbare Zustände weggelassen sind (siehe dazu auch Abbildung 4.4):

- $Q_{12} = \{(q_{01}, q_{02}), (q_{01}, q_{12}), (q_{11}, q_{22})\}$
- $I_{12} = \emptyset$
- $O_{12} = \{a, b\}$
- $\delta_{12} = \{((q_{01}, q_{02}), a, (q_{11}, q_{22})), ((q_{01}, q_{12}), a, (q_{11}, q_{22})), ((q_{01}, q_{02}), b, (q_{01}, q_{12})), ((q_{11}, q_{22}), b, (q_{11}, q_{22}))\}$
- $E_{12} = \emptyset$
- $Qui_{12} = \emptyset$

Es wäre möglich gewesen, die Menge der Ruhe-Zustände bereits anders zu definieren. Dadurch, dass die τ -Transitionen verboten wurden, ist man einer Fallunterscheidung entgangen. Da jedoch Quiescents als Deadlock-Zustände anzusehen sind, aus denen das System ohne Hilfe nicht mehr heraus kommt, müsste hier eigentlich darauf achten, wohin die τ -Transitionen führen und dann möglicherweise die Zustände trotz dieser Transitionen in Menge aufnehmen. Ein Zustand, der keine Outputs machen kann und als einzige τ -Transition eine Schlinge auf sich selbst hat, kann ebenfalls nicht ohne Inputs sich aus diesem Zustand heraus bewegen. Dieser Zustand wird jedoch nach der Definition nicht als ruhig angesehen. Jedoch ist so ein Zustand divergent, da er eine unendliche Folge an internen Aktionen ausführen kann. Dies wurde in [CJK13] mit untersucht. Auch bei anderen Divergenz-Möglichkeiten eines Zustandes, bei denen nicht die Möglichkeit besteht von einem durch τ erreichbaren Zustand aus einen Output zu machen, sollte der Zustand als ruhig angesehen werden. Die passende Definition würde dann wie folgt lauten.

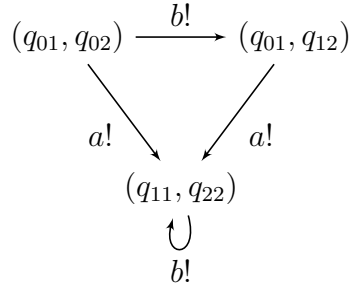


Abbildung 4.4: S_{12}

Definition 4.14 (*Ruhe Alternative*). Ein Ruhe-Zustand ist ein Zustand in einem EIO, der keine Möglichkeit hat ohne einen Input von außen je wieder einen Output zu machen.

Somit ist die Menge der Ruhe-Zustände in einem EIO wie folgt formal definiert: $Qui := \left\{ q \in Q \mid \forall a \in O : q \not\stackrel{a}{\rightarrow} \right\}$.

Diese Art der Definition hätte jedoch die Betrachtung deutlich aufwendiger gemacht und soll deshalb hier nicht behandelt werden.

Literaturverzeichnis

- [BV14] Ferenc Bujtor und Walter Vogler, *Error-Pruning in Interface Automata*, preprint, Universität Augsburg, 2014.
- [CJK13] Chris Chilton, Bengt Jonsson, und Marta Z. Kwiatkowska, *An Algebraic Theory of Interface Automata*, preprint, University of Oxford, 2013.
- [Lyn96] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- [Sch12] Christoph Franz Schlosser, *EIO-Automaten mit Parallelkomposition ohne Internalisierung*, Bachelorarbeit, Universität Augsburg, 2012.