REIHE INFORMATIK
7/99

On Syntactic Action Refinement and Logic

Mila Majster-Cederbaum, Frank Salger
Universität Mannheim
Seminargebäude D 7, 27
D-68131 Mannheim

On Syntactic Action Refinement and Logic

Mila Majster-Cederbaum, Frank Salger

Universität Mannheim
Fakultät für Mathematik und Informatik,
D7, 27, 68131 Mannheim, Germany
{mcb, fsalger}@pi2.informatik.uni-mannheim.de
FAX:++49/621/292-5364

Abstract. Action refinement is a useful methodology for the development of concurrent processes in a stepwise manner. We are here interested in establishing a connection between syntactic action refinement and logic. In the syntactic approach to action refinement, reduction functions are used to remove the refinement operators from process-algebraic expressions thereby providing semantics for them. We incorporate a syntactic action refinement operator to the Hennessy-Milner-Logic and define a logical reduction function for this extended logic. This provides a possibility to refine a process expression and a formula simultaneously on the syntactic level, while preserving their satisfaction relation. It turns out that the assertion $P \models \varphi \Leftrightarrow P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$ where $[a \rightsquigarrow Q]$ denotes the refinement operator both, on process terms and formulas holds in the considered framework under weak and reasonable restrictions.

1 Introduction

The formal development of concurrent systems can be carried out in *process calculi* like CCS [Mil80] or CSP [Hoa90]. These calculi provide operators which are used to compose process expressions to more complex systems (which are also called process expressions). The basic building blocks in such calculi are uninterpreted *atomic actions* which can be seen as the conceptual entities on a chosen level of abstraction. A designer might be interested in providing a more precise or detailed structure for an action which occurs in a given process expression. This is the point where action refinement comes in. Given a process expression P one refines an atomic action a of P by a more complex process expression Q, gaining a more concrete process expression $P[a \rightsquigarrow Q]$. Since the refinement takes place on the process expression itself, it is called *syntactic action refinement*. On the other hand it is possible to define a refinement operator on the semantic level (see e.g. [GGR94]). The application of this operator is called *semantic action refinement*.

Much work has already been done to clarify the properties of semantic and syntactic action refinement (e.g. [AH91,DD93,BDE93,GGR94]), but investigations of action refinement in logical calculi are rare (cf. [Huh96] for a semantic

approach).

Logical calculi, like temporal and modal logics (e.g. [Koz83,Sti87,Eme90,Pen91]) provide the ability to formalize properties of processes. Such properties are denoted by formulas φ of the considered logic. If a process P has the property φ we write $P \models \varphi$. For example the process expression

$$read\_data; send\_data; stop\_transmission$$

determines the process P shown in Figure 1. P obviously satisfies the specification $\varphi \stackrel{def}{=} \langle read\_data \rangle \langle send\_data \rangle \langle stop\_transmission \rangle \top$ in the Hennessy-Milner-Logic.

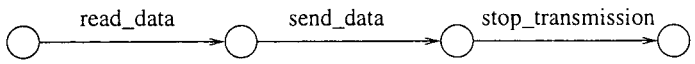

Fig. 1. The Process P

In a later design step two different channels are provided to send data. Hence the action $a = send\_data$ is refined by the process $Q = (send\_data\_channel1 + send\_data\_channel2)$ where '+' denotes nondeterministic choice which gives the process $P[a \rightsquigarrow Q]$ shown in Figure 2. Obviously $P[a \rightsquigarrow Q] \not\models \varphi$, but on the other

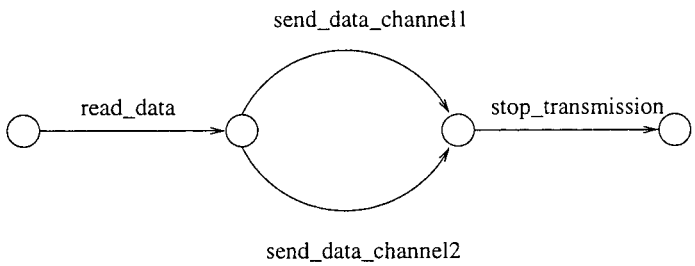

Fig. 2. The Process $P[a \rightsquigarrow Q]$

hand $P[a \rightsquigarrow Q] \models \psi$ where

$$\psi \stackrel{def}{=} \langle read\_data \rangle \langle send\_data\_channel1 \rangle \langle stop\_transmission \rangle \top \wedge \langle read\_data \rangle \langle send\_data\_channel2 \rangle \langle stop\_transmission \rangle \top.$$

We consider the question how ψ can be obtained from φ via a suitable refinement operation $\cdot[a \rightsquigarrow \ldots]$ on the logic i.e. we consider the following question: is it possible to define refinement operators on the process- and the logical calculus

such that we may conclude $P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$? That means, can we simultaneously refine the process expression P and the formula φ, while preserving the satisfaction relation? And, to go a step further: Can we also infer $P \models \varphi$ from $P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$? As we will see, the solution of the above task comes down to the task of defining appropriate reduction functions for the process calculus and the logical calculus. These functions are responsible for merging different abstraction levels and thereby removing the refinement operators.

Our framework consists of a subset of CCS [Mil80] enriched with an operator for action refinement. Action-prefixing is replaced by an operator for sequential composition of process expressions, which is more adequate when dealing with action refinement. To specify properties of processes we use the negation free form of the Hennessy-Milner-Logic advocated in [Mil80], also enriched with an operator for action refinement. The application of this operator will be called *logical action refinement*. We provide a link between syntactic action refinement for process expressions and syntactic action refinement in our adopted Hennessy-Milner-Logic. More concrete, we investigate restrictions on processes P, Q and formulas φ which guarantee

$$P \models \varphi \Leftrightarrow P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q] \tag{1}$$

where a is taken from a fixed set of atomic actions.

The validity of (1) has an interesting implication: Process equivalences are used to identify process expressions. Semantic models based on interleaving are not considered to be 'good' models for action refinement since the used equivalences fail to be congruences for action refinement (cf. [vGG89]). Our approach suggest a revision of this view: Let P be a process and let Φ be a set of formulas that P satisfies based on the interleaving semantics. If P' satisfies Φ as well (e.g. if P' is bisimular to P) then $P[a \rightsquigarrow Q]$ and $P'[a \rightsquigarrow Q]$ both satisfy the set of formulas $\{\varphi[a \rightsquigarrow Q] : \varphi \in \Phi\}$. This result can be used by a system designer, who is not interested in full equality of refined process expressions modulo bisimulation equivalence, but in the fact, that two refined processes both satisfy a refined property (or a set of refined properties) of special interest.

Although verification is usually expressed in terms of implementation relations (cf. [BBR90] for an overview) and action refinement does not support this kind of verification our approach supports 'a priory'-verification with respect to transition behaviour formalized by formulas.

In Section 2 we introduce the process calculus $RBPP$ which contains an operator for syntactic refinement. The concept of logical refinement for the logic $RHML$ is defined in Section 3. Section 4 provides the link between those two concepts, formalized by Theorem 2. We restate the obtained results in the environment of syntactic action refinement in Section 5. A discussion and applications of the results are given in Section 6.

2 The Set *RBPP* of Processes

We fix a set $Act := \{a, b, \ldots\} \cup \{\bar{a}, \bar{b}, \ldots\}$ of actions and a set $Var_{Act} := \{v_1, v_2, \ldots\}$ of distinguished action variables where $Act \cap Var_{Act} = \emptyset$. Action variables from Var_{Act} can be regarded as dummy actions used in what follows as 'place-holders' for processes. We let range $\alpha, \beta, \gamma, \ldots$ over the set $\mathcal{A} := Act \cup Var_{Act}$. Elements of this set are called *atomic performances* or simply *performances*. An action a and its complement $\bar{a}$ may synchronise, performing the distinguished action τ if they are composed in an appropriate way by parallel composition.

2.1 Syntax

Definition 1. *As usual the process expression* 0 *is used to denote a process which is unable to perform any atomic performance.*

1) Let RBP be the language of process expressions generated by the grammar

$$P ::= \alpha \mid (P + P) \mid (P; P) \mid P[\alpha \rightsquigarrow P]$$

2) Let RBPP be the language of process expressions generated by the grammar

$$P ::= 0 \mid \alpha \mid (P + P) \mid (P; P) \mid P[\alpha \rightsquigarrow Q] \mid (P|P)$$

where $Q \in RBP$.

3) Let BPP, BP be the languages of process expressions generated by the grammars for RBPP, RBP respectively, without the rule $P ::= P[\alpha \rightsquigarrow Q]$.

□

Please note that we excluded the silent action τ from the set $\mathcal{A}$. Hence it is not possible to generate process expressions containing τ. This decision is debatable but in our opinion τ-actions are internal knowledge of the system and should be generated only by the system itself via communication. Hence it seems not to be reasonable to supply the ability to specify the systems internal behaviour since a designer is not explicitly informed about it. Moreover without knowledge about the internal behaviour of a system it seems to be unnatural to be able to refine it by known, observable actions (see [AH91]). Only the system itself should know about its internal actions and no refine-methodology should affect them.

As in [GGR94] we define a function $\xi : BPP \to 2^{\mathcal{A}}$ which gives the set of performances of a process expression.

Definition 2. *Let* $P, Q \in BPP$ *be process expressions. We define the function* $\xi : BPP \to 2^{\mathcal{A}}$ *as follows:*

$\xi(0) := \emptyset$

$\xi(\alpha) := \{\alpha\}$

$\xi((P * Q)) := \xi(P) \cup \xi(Q) \quad$ *if* $* \in \{+, ;, |\}$
□

The function $\xi_{Act} : BPP \to 2^{Act}$ is defined in analogy to the function ξ.

2.2 The Reduction Function for *RBPP*

We now proceed to the definition of performance refinement. As in [AH91] we use a *reduction function* $red : RBPP \to BPP$ which removes the occurrence of all refinement operators in a process expression. The function *red* uses syntactic substitution defined as in [GGR94].

Definition 3. *Let $P, P_1, P_2, Q \in BPP$ be process expressions.* Syntactic substitution, *denoted $(P)\{Q/\alpha\}$ is defined as follows:*

$(0)\{Q/\alpha\} := 0$

$(\alpha)\{Q/\beta\} := \begin{cases} Q \text{ if } \alpha = \beta \\ \alpha \text{ otherwise} \end{cases}$

$((P_1 * P_2))\{Q/\alpha\} := ((P_1)\{Q/\alpha\} * (P_2)\{Q/\alpha\})$ *where* $* \in \{+, ;, |\}$
□

Remark 1. To avoid excessive use of brackets we sometimes use the notation $P\{Q/\alpha\}$ instead of $(P)\{Q/\alpha\}$ if the context avoids ambiguity. □

The following remark shows that several nested applications of the substitution operation can be reduced to only one such application. It is proved by induction on the structure of $P \in BPP$.

Remark 2. Let $P, Q_1, Q_2 \in BPP$ be process expressions and $\gamma_1, \gamma_2 \in \mathcal{A}$. Further let $* \in \{;, +, |\}$. If $\gamma_1, \gamma_2 \notin \xi((Q_1 * Q_2)) \cup \xi(P)$ and $\gamma_1 \neq \gamma_2$ then $(((P)\{(\gamma_1 * \gamma_2)/\alpha\})\{Q_2/\gamma_2\})\{Q_1/\gamma_1\} = (P)\{(Q_1 * Q_2)/\alpha\}$. □

Definition 4. *Let $P, P_1, P_2 \in RBPP$ and $Q \in RBP$ be process expressions. The function $red : RBPP \to BPP$ is defined as follows:*

$red(0) := 0$

$red(\alpha) := \alpha$

$red((P_1 * P_2)) := (red(P_1) * red(P_2))$ *where* $* \in \{+, ;, |\}$

$red(P[\alpha \rightsquigarrow Q]) := (red(P))\{red(Q)/\alpha\}$
□

Remark 3. Please note that the refinement of an action might suspend a potential communication. We could use instead the 'hybrid' operator $\cdot[a \rightsquigarrow Q]_H := (\cdot[a \rightsquigarrow Q])[\bar{a} \rightsquigarrow \bar{Q}]$ if this is felt to be an undue property of a refinement operator. $\bar{Q}$ is the expression Q where every action is replaced by its complement (cf. [AH91]). □

Remark 4 states that one application of the reduction function is sufficient to remove all refinement operators occurring in a process expression.

Remark 4. Let $P \in RBPP$ and $Q \in RBP$. Then $red(P[\alpha \rightsquigarrow Q]) = red(red(P)[\alpha \rightsquigarrow Q])$. □

Lemma 1. *Let $P \in RBPP$, $Q_1, Q_2 \in RBP$ be process expressions and $\gamma_1, \gamma_2 \in \mathcal{A}$ such that $\gamma_1 \neq \gamma_2$. Further let $* \in \{\ ;\ ,\ +\ \}$. If $\gamma_1, \gamma_2 \notin \xi(P) \cup \xi(red(Q_1; Q_2))$ then $red(((P[\alpha \rightsquigarrow (\gamma_1 * \gamma_2)])[\gamma_2 \rightsquigarrow Q_2])[\gamma_1 \rightsquigarrow Q_1]) = red(P[\alpha \rightsquigarrow (Q_1 * Q_2)])$.*

Proof. Follows by Remark 2 and Remark 4.
■

In the presence of the sequential composition operator ';' it is common to work with a special predicate $\surd$ indicating successful termination (cf. [AH91]).

Definition 5. *Let $\surd \subset RBPP$ be the least set closed under the rules*

$0 \in \surd$

$E \in \surd$ *and* $F \in \surd \Rightarrow (E * F) \in \surd$ *where* $* \in \{|, +, ;\}$

$E \in \surd \Rightarrow E[\alpha \rightsquigarrow Q] \in \surd$
□

2.3 The Semantics of *RBPP*

We now define the operational semantics of the language $RBPP$.

Definition 6. *Let $E, E_1, E_2, F \in RBPP$ and $Q \in RBP$ be process expressions.*

$$transition: \quad \frac{}{\alpha \xrightarrow{\alpha} 0}$$

$$choice_1: \quad \frac{E \xrightarrow{\alpha} E'}{(E+F) \xrightarrow{\alpha} E'}$$

$$choice_2: \quad \frac{F \xrightarrow{\alpha} F'}{(E+F) \xrightarrow{\alpha} F'}$$

$$sequential\ composition_1: \quad \frac{E \xrightarrow{\alpha} E'}{(E;F) \xrightarrow{\alpha} (E';F)}$$

$$sequential\ composition_2: \quad \frac{F \xrightarrow{\alpha} F'}{(E;F) \xrightarrow{\alpha} F'} \quad if\ E \in \surd$$

$$parallel\ composition_1: \quad \frac{E_1 \xrightarrow{\alpha} E_1'}{(E_1|E_2) \xrightarrow{\alpha} (E_1'|E_2)}$$

$$parallel\ composition_2: \quad \frac{E_2 \xrightarrow{\alpha} E_2'}{(E_1|E_2) \xrightarrow{\alpha} (E_1|E_2')}$$

$$parallel\ composition_3: \quad \frac{E_1 \xrightarrow{\alpha} E_1' \quad E_2 \xrightarrow{\bar{\alpha}} E_2'}{(E_1|E_2) \xrightarrow{\tau} (E_1'|E_2')}$$

$$refinement: \quad \frac{red(E[\alpha \rightsquigarrow Q]) \xrightarrow{\alpha} F}{E[\alpha \rightsquigarrow Q] \xrightarrow{\alpha} F}$$

□

The last rule expresses the philosophy that the behaviour of the process P is considered to be identical to that of the process $red(P)$ (cf. [AH91]).

A process expression P determines a *labelled transition system with termination* i.e. a tupel $(P, RBPP, \mathcal{A}, \rightarrow, \surd)$ where $P \in RBPP$ is the initial state and $\rightarrow \subseteq RBPP \times \mathcal{A} \cup \{\tau\} \times RBPP$ is the set of transitions, derived from the operational semantics. $\surd$ gives the set of terminated states. A process expression P' is called an α-sucessor of P iff $P \xrightarrow{\alpha} P'$. P' is called a sucessor if there is a performance α s.t. P' is an α-sucessor of P. Sometimes we use the notation $P \xrightarrow{\alpha}$ to indicate the ability of P to execute an α-performance.

Some elementary properties of the function red are summarized in the following which allow us to relate the behaviour of P and $red(P[\alpha \rightsquigarrow Q])$. First we show that refinements behave well in the sense that they neither remove a process expression from the set $\surd$ of terminated states nor introduce a reduced process expression to it.

Lemma 2. *Let $P \in BPP$ and $Q \in RBP$. Then $P \in \surd$ iff $red(P[\alpha \rightsquigarrow Q]) \in \surd$.*

Proof. Immediate.

Lemma 3. *Let $P \in RBPP$. Then $P \in \surd$ iff $red(P) \in \surd$.*

Proof. Follows from Definition 6, the inability of $P \in \surd$ to commit any performance $\alpha \in \mathcal{A}$ and the fact that $Q \in RBP$ for any term of the from $P[\alpha \rightsquigarrow Q]$. ■

Lemma 4. *Let $P \in BPP$ and $Q \in RBP$ be process expressions. If $\alpha \neq \beta$ then*

1) $\forall P \forall P'(P \xrightarrow{\beta} P' \Rightarrow red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q]))$

2) If $\beta \notin \xi(red(Q))$ *then* $\forall P \forall P' \Big(red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P' \Rightarrow \exists P''(P \xrightarrow{\beta} P''$ *and* $red(P''[\alpha \rightsquigarrow Q]) = P') \Big)$

Proof. 1) By structural induction on $P \in BPP$.

$P = 0$ and $P = \gamma$ where $\gamma \neq \beta$: Trivial.

$P = \beta$:

Obvious as $P \xrightarrow{\beta} 0$, $red(P[\alpha \rightsquigarrow Q]) = \beta = P$ since $\alpha \neq \beta$ and $red(0[\alpha \rightsquigarrow Q]) = 0$.

$P = (P_1 + P_2)$:

Assume $(P_1 + P_2) \xrightarrow{\beta} P'$ for some $P' \in BPP$. By Definition 6 we get $P_1 \xrightarrow{\beta} P'$ or $P_2 \xrightarrow{\beta} P'$. W.l.o.g. assume the former. Then

$$red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q])$$

by the induction hypothesis. This implies

$$(red(P_1[\alpha \rightsquigarrow Q]) + red(P_2[\alpha \rightsquigarrow Q])) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q])$$

As $(red(P_1[\alpha \rightsquigarrow Q]) + red(P_2[\alpha \rightsquigarrow Q])) = red((P_1 + P_2)[\alpha \rightsquigarrow Q])$ we get

$$red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q]).$$

$P = (P_1|P_2)$:

Assume $(P_1|P_2) \xrightarrow{\beta} P'$ for some $P' \in BPP$. Since $\beta \neq \tau$ (remember $\tau \notin \mathcal{A}$) we have $P' = P_1'|P_2$ where $P_1 \xrightarrow{\beta} P_1'$ or $P' = P_1|P_2'$ where $P_2 \xrightarrow{\beta} P_2'$. W.l.o.g. assume the former. By the induction hypothesis we get $red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P_1'[\alpha \rightsquigarrow Q])$. This implies

$$(red(P_1[\alpha \rightsquigarrow Q])|\bar{P}) \xrightarrow{\beta} (red(P_1'[\alpha \rightsquigarrow Q])|\bar{P})$$

for any $\tilde{P} \in BPP$. Now let $\tilde{P} = red(P_2[\alpha \rightsquigarrow Q])$. Then

$$red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red((P_1'|P_2))[\alpha \rightsquigarrow Q]).$$

$\underline{P = (P_1; P_2)}$:

Assume $(P_1; P_2) \xrightarrow{\beta} P'$ for some $P' \in BPP$. By Definition 6 we have to consider two cases:

Case 1: $P_1 \in \surd$. By the assumption we must have $P_2 \xrightarrow{\beta} P'$. We get

$$red(P_2[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q])$$

by the induction hypothesis. By Lemma 2 we get $red(P_1[\alpha \rightsquigarrow Q]) \in \surd$. This implies

$$(red(P_1[\alpha \rightsquigarrow Q]); red(P_2[\alpha \rightsquigarrow Q])) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q])$$

hence

$$red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow Q]).$$

Case 2: $P_1 \notin \surd$. Then $P_1 \xrightarrow{\beta} P_1'$ and $(P_1; P_2) \xrightarrow{\beta} (P_1'; P_2)$ by Definition 6. We get

$$red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(P_1'[\alpha \rightsquigarrow Q])$$

by the induction hypothesis. By Definition 6 we obtain

$$(red(P_1[\alpha \rightsquigarrow Q]); \tilde{P}) \xrightarrow{\beta} (red(P_1'[\alpha \rightsquigarrow Q]); \tilde{P})$$

for any $\tilde{P} \in BPP$. Let $\tilde{P} = red(P_2[\alpha \rightsquigarrow Q])$ then

$$red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red((P_1'; P_2)[\alpha \rightsquigarrow Q]) = red(P'[\alpha \rightsquigarrow Q]).$$

2) By structural induction on $P \in BPP$.

$\underline{P = 0}$: Trivial, since $red(P[\alpha \rightsquigarrow Q]) = 0$.

$\underline{P = \gamma}$:

Case 1: $\gamma = \alpha$. Then $red(\alpha[\alpha \rightsquigarrow Q]) = red(Q)$. But $\beta \notin \xi(red(Q))$ by the condition whence the implication is trivially true.
Case 2: $\gamma = \beta$. Then $red(\beta[\alpha \rightsquigarrow Q]) = \beta$ since $\alpha \neq \beta$. Hence $P' = 0$. We choose $P'' = 0$ whence $red(P''[\alpha \rightsquigarrow Q]) = 0 = P'$.
Case 3: $\gamma \neq \alpha$ and $\gamma \neq \beta$. Then $red(P[\alpha \rightsquigarrow Q]) = \gamma$. Since $\gamma \neq \beta$ the implication is trivially true.

$\underline{P = (P_1 + P_2)}$:

Assume $red((P_1 + P_2)[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P'$ for some $P' \in BPP$. From Definition 3 and Definition 4 we obtain

$$(red(P_1[\alpha \rightsquigarrow Q]) + red(P_2[\alpha \rightsquigarrow Q])) \xrightarrow{\beta} P'$$

hence

$$red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P' \text{ or } red(P_2[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P'.$$

W.l.o.g. assume the former. Then

$$\exists P''(P_1 \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = P')$$

and

$$\exists P''((P_1 + P_2) \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = P').$$

$\underline{P = (P_1|P_2)}$:

Assume $red((P_1|P_2)[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P'$ for some $P' \in BPP$. Then we get

$$(red(P_1[\alpha \rightsquigarrow Q])|red(P_2[\alpha \rightsquigarrow Q])) \xrightarrow{\beta} P'.$$

Since $\beta \neq \tau$ ($\tau \notin \mathcal{A}$) we get

$$P' = (E|red(P_2[\alpha \rightsquigarrow Q])) \text{ where } red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} E \quad E \in BPP$$

or

$$P' = (red(P_1[\alpha \rightsquigarrow Q])|F) \text{ where } red(P_2[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} F \quad F \in BPP.$$

W.l.o.g. assume the former. Then

$$\exists \tilde{P}(P_1 \xrightarrow{\beta} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow Q]) = E)$$

and

$$\exists \tilde{P}((P_1|P_2) \xrightarrow{\beta} (\tilde{P}|P_2) \text{ and } red(\tilde{P}[\alpha \rightsquigarrow Q]) = E) \quad (*)$$

As

$$red((\tilde{P}|P_2)[\alpha \rightsquigarrow Q]) = (red(\tilde{P}[\alpha \rightsquigarrow Q])|red(P_2[\alpha \rightsquigarrow Q]))$$

we get

$$red((\tilde{P}|P_2)[\alpha \rightsquigarrow Q]) = (E|red(P_2[\alpha \rightsquigarrow Q])) = P'$$

from $(*)$. Hence we conclude

$$\exists P''((P_1|P_2) \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = P')$$

by choosing $P'' = (\tilde{P}|P_2)$.

$P = (P_1; P_2)$:

Assume $red((P_1; P_2)[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} P'$ for some $P' \in BPP$. We obtain

$$(red(P_1[\alpha \rightsquigarrow Q]); red(P_2[\alpha \rightsquigarrow Q])) \xrightarrow{\beta} P'.$$

Case 1: $P' = (E; red(P_2[\alpha \rightsquigarrow Q]))$ where $red(P_1[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} E$.

Case 2: $P' = F$ where $red(P_2[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} F$ and $red(P_1[\alpha \rightsquigarrow Q]) \in \surd$.

For case 1 we get

$$\exists \tilde{P}(P_1 \xrightarrow{\beta} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow Q]) = E) \quad (*)$$

by the induction hypothesis. This implies

$$\exists \tilde{P}((P_1; P_2) \xrightarrow{\beta} (\tilde{P}; P_2) \text{ and } red(\tilde{P}[\alpha \rightsquigarrow Q]) = E)$$

by Definition 6. As

$$red((\tilde{P}; P_2)[\alpha \rightsquigarrow Q]) = (E; red(P_2[\alpha \rightsquigarrow Q])) = P'$$

by assertion $(*)$, we conclude

$$\exists P''((P_1; P_2) \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = P')$$

by choosing $P'' = (\tilde{P}; P_2)$.

For case 2 we obtain

$$\exists P''(P_2 \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = F)$$

by the induction hypothesis. By Lemma 2 we have $P_1 \in \surd$ since $red(P_1[\alpha \rightsquigarrow Q]) \in \surd$. Hence we obtain

$$\exists P''((P_1; P_2) \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = F).$$

■

Lemma 5. *Let $P \in BPP$ be a process expression. Then*

1) $\forall P \forall P'(P \xrightarrow{\alpha} P' \Rightarrow red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} red(P'[\alpha \rightsquigarrow \beta]))$

2) If $\beta \notin \xi(P)$ *then* $\forall P \forall P' \Big(red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} P' \Rightarrow \exists P''(P \xrightarrow{\alpha} P''$ *and* $red(P''[\alpha \rightsquigarrow \beta]) = P' \Big)$

Proof. Assertion 1) and 2) are proved by structural induction on $P \in BPP$. We only show the cases where the proof differs from the proof of Lemma 4.

For assertion 1) we show

$P = \alpha$:

$\alpha \xrightarrow{\alpha} 0$, $red(P[\alpha \rightsquigarrow \beta]) = \beta$ whence $red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} 0$ which completes this case since $red(0[\alpha \rightsquigarrow \beta]) = 0$.

For assertion 2) we show

$P = \gamma$:

Case 1: $\gamma = \alpha$. Then $red(\alpha[\alpha \rightsquigarrow \beta]) = \beta$. By Definition 6 we have $P' = 0$. We choose $P'' = 0$. Then $P \xrightarrow{\alpha} P''$ and $red(P''[\alpha \rightsquigarrow \beta]) = red(0[\alpha \rightsquigarrow \beta]) = 0 = P'$.

Case 2: $\gamma = \beta$: cannot occur due to the condition $\beta \notin \xi(P)$ of the lemma.

Case 3: $\gamma \neq \alpha$ and $\gamma \neq \beta$: The implication is trivially true.
■

Lemma 6. *Let $P \in BPP$ be a process expression and $\gamma_1, \gamma_2 \in \mathcal{A}$. Then*

1) $\forall P \forall P' \Big(P \xrightarrow{\alpha} P' \Rightarrow$

$\exists P''(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big)$

2) If $\gamma_1, \gamma_2 \notin \xi(P)$ *then* $\forall P \forall P' \forall P'' \Big(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P'' \Rightarrow$

$\exists \tilde{P}(P \xrightarrow{\alpha} \tilde{P}$ *and* $red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'')\Big)$

Proof. 1) by induction on the structure of $P \in BPP$.

$P = 0$: Trivial.

$P = \gamma$:

Case 1: $\gamma \neq \alpha$. Then the implication is trivially true.

Case 2: $\gamma = \alpha$. Then $\alpha \xrightarrow{\alpha} 0$ whence $P' = 0$. Now $red(\alpha[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = (\gamma_1;\gamma_2)$. We choose $P'' = (0;\gamma_2)$. Then

$$(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])).$$

$\underline{P = (P_1 + P_2)}$:

Assume $(P_1 + P_2) \xrightarrow{\alpha} P'$ for some $P' \in BPP$. Then we obtain

$$P_1 \xrightarrow{\alpha} P' \text{ or } P_2 \xrightarrow{\alpha} P'$$

by Definition 6. W.l.o.g. assume the former. Then

$$\exists P''(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])).$$

This implies

$$\exists P''(red((P_1 + P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])).$$

$\underline{P = (P_1|P_2)}$:

Assume $(P_1|P_2) \xrightarrow{\alpha} P'$ for some $P' \in BPP$. Since $\alpha \neq \tau$ ($\tau \notin \mathcal{A}$) we have

$$P' = (P_1'|P_2) \text{ where } P_1 \xrightarrow{\alpha} P_1'$$

or

$$P' = (P_1|P_2') \text{ where } P_2 \xrightarrow{\alpha} P_2'$$

by Definition 6. W.l.o.g. assume the former. Then

$$\exists P''(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P_1'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

by the induction hypothesis. This implies

$$\exists P''\Big((red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \xrightarrow{\gamma_1} (P''|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$$

and

$$(P''|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_2} (red(P_1'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big).$$

We obtain

$$\exists P''\Big(red((P_1|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} (P''|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$$

and

$$(P''|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_2} red((P_1'|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big).$$

We conclude

$$\exists \tilde{P}(red((P_1|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} \tilde{P} \xrightarrow{\gamma_2} red((P_1'|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

by choosing $\tilde{P} = (P''|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$.

$P = (P_1;P_2)$:

Assume $(P_1;P_2) \xrightarrow{\alpha} P'$ for some $P' \in BPP$.

Case 1: $P' = (P_1';P_2)$ where $P_1 \xrightarrow{\alpha} P_1'$

Case 2: $P' = P_2'$ where $P_2 \xrightarrow{\alpha} P_2'$ and $P_1 \in \surd$

For case 1 we obtain

$$\exists P''(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P_1'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

by the induction hypothesis. This implies

$$\exists P''\Big((red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]);red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \xrightarrow{\gamma_1} (P'';red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

and

$$(P'';red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_2} (red(P_1'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]);red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big).$$

We obtain

$$\exists P''\Big(red((P_1;P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} (P'';red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \text{ and}$$

$$(P'';red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \xrightarrow{\gamma_2} red((P_1';P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big)$$

by Definition 3 and Definition 4. Hence we conclude the desired result

$$\exists \tilde{P}(red((P_1;P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} \tilde{P} \xrightarrow{\gamma_2} red((P_1';P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

by choosing $\tilde{P} = (P'';red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$.

For case 2 we have

$$\exists P''(red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P_2'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

by the induction hypothesis. Since $P_1 \in \surd$ we have $red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \in \surd$ by Lemma 2. Hence

$$\exists P''((red((P_1;P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P_2'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))).$$

2) By induction on the structure of $P \in BPP$.

$P = 0$: Trivial, since $red(0[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = 0$.

$P = \gamma$:

Case 1: $\gamma \neq \alpha$. The $red(\gamma[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = \gamma$. Hence the implication is trivially true since $\gamma_1, \gamma_2 \notin \xi(P)$ by the condition of assertion 2).

Case 2: $\gamma = \alpha$. Then $red(\alpha[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = (\gamma_1;\gamma_2)$. Hence we have $P' = (0;\gamma_2)$ and $P'' = 0$. On the other hand $\alpha \xrightarrow{\alpha} 0$. We choose $\tilde{P} = 0$. Hence

$$P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P''.$$

$P = (P_1 + P_2)$:

Assume $red((P_1 + P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''$ for $P', P'' \in BPP$. We obtain

$$(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) + red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''.$$

Hence we obtain

$$red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''$$

or

$$red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''.$$

W.l.o.g. assume the former. Then

$$\exists \tilde{P}(P_1 \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'')$$

by the induction hypothesis. This implies

$$\exists \tilde{P}((P_1 + P_2) \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'').$$

$P = (P_1|P_2)$:

Assume $red((P_1|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''$ for $P', P'' \in BPP$. We obtain

$$(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \xrightarrow{\gamma_1} P' \xrightarrow{\gamma_2} P''.$$

Since $\gamma_1, \gamma_2 \neq \tau$ ($\tau \notin \mathcal{A}$) we have one of the following cases:

Case 1)

$$P' = (E|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \text{ where } red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} E$$

and

$$P'' = (E'|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \text{ where } E \xrightarrow{\gamma_2} E'$$

(Note that $red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_2}{\not\rightarrow}$ since $\gamma_2 \notin \xi(P)$)

Case 2)

$$P' = (red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|F) \text{ where } red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_1}{\rightarrow} F$$

and

$$P'' = (red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|F') \text{ where } F \stackrel{\gamma_2}{\rightarrow} F'$$

(Note that $red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_2}{\not\rightarrow}$ since $\gamma_2 \notin \xi(P)$)

We consider case 1), i.e. we have

$$red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_1}{\rightarrow} E \stackrel{\gamma_2}{\rightarrow} E'$$

whence we get

$$\exists \tilde{P}(P_1 \stackrel{\alpha}{\rightarrow} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = E').$$

This implies

$$\exists \tilde{P}((P_1|P_2) \stackrel{\alpha}{\rightarrow} (\tilde{P}|P_2)) \quad (*)$$

and

$$(red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) = (E'|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]))$$

which gives

$$red((\tilde{P}|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = (E'|red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) = P'' \quad (**)$$

Taking $(*)$ and $(**)$ together we obtain

$$\exists \tilde{P}((P_1|P_2) \stackrel{\alpha}{\rightarrow} (\tilde{P}|P_2) \text{ and } red((\tilde{P}|P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'')$$

whence

$$\exists \tilde{P}((P_1|P_2) \stackrel{\alpha}{\rightarrow} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'').$$

$P = (P_1;P_2)$:

Assume $red((P_1;P_2)[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_1}{\rightarrow} P' \stackrel{\gamma_2}{\rightarrow} P''$ for $P', P'' \in BPP$. We obtain

$$(red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]);red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \stackrel{\gamma_1}{\rightarrow} P' \stackrel{\gamma_2}{\rightarrow} P''.$$

Case 1)

$$P' = (E;red(P_2[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \text{ where } red(P_1[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \stackrel{\gamma_1}{\rightarrow} E$$

and

$$P'' = (E'; red(P_2[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])) \text{ where } E \xrightarrow{\gamma_3} E'$$

(Note that $red(P_1[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \xrightarrow{\gamma_1}$ implies $E \notin \surd$ since $\gamma_1 \notin \xi(P)$)

Case 2)

$$P' = F \text{ where } red(P_2[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \xrightarrow{\gamma_1} F \text{ and } red(P_1[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \in \surd$$

and

$$P'' = F' \text{ where } F \xrightarrow{\gamma_3} F'$$

For case 1 we have

$$red(P_1[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \xrightarrow{\gamma_1} E \xrightarrow{\gamma_3} E'$$

whence we get

$$\exists \tilde{P}(P_1 \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = E')$$

by the induction hypothesis. This implies

$$\exists \tilde{P}((P_1; P_2) \xrightarrow{\alpha} (\tilde{P}; P_2)) \quad (*)$$

and

$$(red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]); red(P_2[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])) = (E'; red(P_2[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]))$$

which gives

$$red((\tilde{P}; P_2)[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = (E'; red(P_2[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])) = P'' \quad (**)$$

Taking $(*)$ and $(**)$ together we obtain

$$\exists \tilde{P}((P_1; P_2) \xrightarrow{\alpha} (\tilde{P}; P_2) \text{ and } red((\tilde{P}; P_2)[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = P'')$$

whence we obtain

$$\exists \tilde{P}((P_1; P_2) \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = P'').$$

For case 2 we obtain

$$\exists \tilde{P}(P_2 \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = F')$$

by the induction hypothesis. From $red(P_1[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \in \surd$ we have $P_1 \in \surd$ by Lemma 2. Hence we obtain

$$\exists \tilde{P}((P_1; P_2) \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = F').$$

■

Lemma 7. *Let $P \in BPP$ and $\gamma_1, \gamma_2 \in \mathcal{A}$. Then*

1) $\forall P \forall P' \Big(P \xrightarrow{\alpha} P' \Rightarrow$

$\forall i \in \{1,2\} \Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} red(P'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \Big) \Big)$

2) If $\gamma_1, \gamma_2 \notin \xi(P)$ *then*

$\forall P \forall P' \Big(\exists i \in \{1,2\} \Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} P' \Rightarrow$

$\exists \tilde{P} (P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) = P') \Big) \Big)$

Proof. Both assertions are proved by induction on the structure of $P \in BPP$.
■

Lemma 8. *Let $P \in RBPP$. Then*
$\forall P \forall P' \in RBPP \big(P \xrightarrow{\alpha} P' \Rightarrow red(P) \xrightarrow{\alpha} red(P') \big)$

Proof. The proof is by induction on the strucure of $P \in RBPP$.

$\underline{P = 0}$: Trivial.

$\underline{P = \alpha}$: Trivial since $red(P) = P$.

$\underline{P = (P_1 + P_2)}$:

Suppose $(P_1 + P_2) \xrightarrow{\alpha} P'$ for some $P' \in RBPP$. Then

$$P_1 \xrightarrow{\alpha} P' \text{ or } P_2 \xrightarrow{\alpha} P'$$

by Definition 6. W.l.o.g. assume the former. Hence

$$red(P_1) \xrightarrow{\alpha} red(P')$$

by the induction hypothesis. This implies

$$(red(P_1) + red(P_2)) \xrightarrow{\alpha} red(P')$$

by Definition 6 whence we conclude

$$red(P_1 + P_2) \xrightarrow{\alpha} red(P')$$

by Definition 4.

$\underline{P = (P_1 | P_2)}$:

Assume $(P_1 | P_2) \xrightarrow{\alpha} P'$ for some $P' \in RBPP$. Since $\beta \neq \tau$ (remember $\tau \notin \mathcal{A}$) we get

$$P' = (E | P_2) \text{ where } P_1 \xrightarrow{\alpha} E \quad E \in BPP$$

or

$$P' = (P_1|F) \text{ where } P_2 \xrightarrow{\alpha} F \quad F \in BPP$$

by Definition 6. W.l.o.g. assume the former. Then

$$red(P_1) \xrightarrow{\alpha} red(E)$$

by the induction hypothesis. This implies

$$(red(P_1)|red(P_2)) \xrightarrow{\alpha} (red(E)|red(P_2))$$

by Definition 6, i.e.

$$(red(P_1)|red(P_2)) \xrightarrow{\alpha} red(P').$$

$\underline{P = (P_1; P_2)}$:

Assume $(P_1; P_2) \xrightarrow{\alpha} P'$ for some $P' \in RBPP$. According to Definition 6 we consider two cases:

Case 1: $P' = (E; P_2)$ where $P_1 \xrightarrow{\alpha} E$.

Case 2: $P' = F$ where $P_2 \xrightarrow{\alpha} F$ and $P_1 \in \surd$.

For case 1 we get

$$red(P_1) \xrightarrow{\alpha} red(E)$$

which implies

$$(red(P_1); red(P_2)) \xrightarrow{\alpha} (red(E); red(P_2))$$

whence

$$red(P_1; P_2) \xrightarrow{\alpha} red(E; P_2).$$

For case 2 we obtain

$$red(P_2) \xrightarrow{\alpha} red(F)$$

by the induction hypothesis. By Lemma 3 we have $red(P_1) \in \surd$ since $P_1 \in \surd$. Hence

$$(red(P_1); red(P_2)) \xrightarrow{\alpha} red(F)$$

i.e. $red(P_1; P_2) \xrightarrow{\alpha} red(F)$.

$\underline{P = \tilde{P}[\alpha \rightsquigarrow Q]}$:

Assume $\tilde{P}[\alpha \rightsquigarrow Q] \xrightarrow{\alpha} E'$ for some $E' \in RBPP$. Then we must have $red(\tilde{P}[\alpha \rightsquigarrow Q]) \xrightarrow{\alpha} E'$ by Definition 6 and $E' = red(E') \in BPP$ which completes this induction step.

■

Remark 5. The converse of Lemma 8 does not hold as can be seen in the following:
Let $P = (a|b)$. Then $red(P) = P$ whence $P \xrightarrow{b} red((c[c \rightsquigarrow a]|0))$. But on the other hand $P \not\xrightarrow{b} (c[c \rightsquigarrow a]|0)$. □

3 The Logic *RHML*

In this section we introduce the logic HML following the line of [Mil80]. We augment HML by an operator for logical action refinement which gives the logic $RHML$.

3.1 Syntax

Definition 7. *HML is the language of assertions generated by the grammar*

$$\Phi ::= \top \mid \bot \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid [\alpha]\Phi \mid \langle\alpha\rangle\Phi$$

where α ranges over $\mathcal{A}$. Let $RHML$ be the language generated by the above grammar with the additional rule

$$\Phi ::= \Phi[\alpha \rightsquigarrow Q]$$

where $Q \in RBP$.
□

Definition 8. *Let $\varphi, \varphi_1, \varphi_2 \in HML$. We define the function $\xi : HML \to 2^{\mathcal{A}}$ as follows:*

$\xi(*) := \emptyset \quad$ *if* $* \in \{\top, \bot\}$

$\xi((\varphi_1 * \varphi_2)) := \xi(\varphi_1) \cup \xi(\varphi_2) \quad$ *if* $* \in \{\wedge, \vee\}$

$\xi([\alpha]\varphi) := \{\alpha\} \cup \xi(\varphi)$

$\xi(\langle\alpha\rangle\varphi) := \{\alpha\} \cup \xi(\varphi)$

□

The function $\xi_{Act} : HML \to 2^{Act}$ is defined analogously to the function ξ.

3.2 The Logical Reduction

We introduce a concept of logical substitution by the way of which we are able to define the reduction of logical formulas.

Definition 9. *Let $Q, Q_1, Q_2 \in BP$ and $\Phi, \Phi_1, \Phi_2 \in HML$ and $n, m \in \mathbb{N}_0$. The operation of* logical substitution, *$(\Phi)\{\alpha \rightsquigarrow Q\}$ is defined as follows:*

$$(*)\{\alpha \rightsquigarrow Q\} := * \quad \text{if } * \in \{\top, \bot\}$$

$$((\Phi_1 * \Phi_2))\{\alpha \rightsquigarrow Q\} := ((\Phi_1)\{\alpha \rightsquigarrow Q\} * (\Phi_2)\{\alpha \rightsquigarrow Q\}) \quad \text{if } * \in \{\wedge, \vee\}$$

$$([\beta]\Phi)\{\alpha \rightsquigarrow Q\} := [\beta](\Phi)\{\alpha \rightsquigarrow Q\} \quad \text{if } \alpha \neq \beta$$

$$([\alpha]\Phi)\{\alpha \rightsquigarrow Q\} :=$$

$$\begin{cases} [\beta](\Phi)\{\alpha \rightsquigarrow Q\} & \text{if } Q = \beta \\ (([v_n](\Phi)\{\alpha \rightsquigarrow Q\})\{v_n \rightsquigarrow Q_1\} \wedge ([v_m](\Phi)\{\alpha \rightsquigarrow Q\})\{v_m \rightsquigarrow Q_2\}) & \text{if } Q = (Q_1 + Q_2) \\ ([v_n]([v_m](\Phi)\{\alpha \rightsquigarrow Q\})\{v_m \rightsquigarrow Q_2\})\{v_n \rightsquigarrow Q_1\} & \text{if } Q = (Q_1; Q_2) \end{cases}$$

$$(\langle\beta\rangle\Phi)\{\alpha \rightsquigarrow Q\} := \langle\beta\rangle(\Phi)\{\alpha \rightsquigarrow Q\} \quad \text{if } \alpha \neq \beta$$

$$(\langle\alpha\rangle\Phi)\{\alpha \rightsquigarrow Q\} :=$$

$$\begin{cases} \langle\beta\rangle(\Phi)\{\alpha \rightsquigarrow Q\} & \text{if } Q = \beta \\ ((\langle v_n\rangle(\Phi)\{\alpha \rightsquigarrow Q\})\{v_n \rightsquigarrow Q_1\} \wedge (\langle v_m\rangle(\Phi)\{\alpha \rightsquigarrow Q\})\{v_m \rightsquigarrow Q_2\}) & \text{if } Q = (Q_1 + Q_2) \\ (\langle v_n\rangle(\langle v_m\rangle(\Phi)\{\alpha \rightsquigarrow Q\})\{v_m \rightsquigarrow Q_2\})\{v_n \rightsquigarrow Q_1\} & \text{if } Q = (Q_1; Q_2) \end{cases}$$

For the introduction of variables v_n, v_m we require $v_n, v_m \notin \xi([\alpha]\Phi) \cup \xi(Q)$ and $n \neq m$ ($v_n, v_m \notin \xi(\langle\alpha\rangle\Phi) \cup \xi(Q)$ respectivly).
□

Remark 6. To avoid excessive use of brackets we sometimes use the notation $\Phi\{\alpha \rightsquigarrow Q\}$ instead of $(\Phi)\{\alpha \rightsquigarrow Q\}$ if the context avoids ambiguity.
□

Example 1. Let $\varphi = [a](\langle b\rangle\top \vee [a]\bot)$ and $Q = (d + e)$ where $a, b, d, e \in \mathcal{A}$. Then

$$(\varphi)\{a \rightsquigarrow Q\} = ([d](\langle b\rangle\top \vee ([d]\bot \wedge [e]\bot)) \wedge [e](\langle b\rangle\top \vee ([d]\bot \wedge [e]\bot)))$$

□

To prove that the operation of logical substitution is always defined we introduce a suitable transitive, anti-symmetrical relation.

Definition 10. *The length* $|\cdot| : RHML \to I\!N$ *of formulas is given by*

$$|*| := 1 \quad * \in \{\bot, \top\}$$

$$|(\varphi_1 * \varphi_2)| := 1 + |\varphi_1| + |\varphi_2| \quad * \in \{\vee, \wedge\}$$

$$|*\varphi| := 1 + |\varphi| \quad * \in \{[\alpha], \langle\alpha\rangle\}$$

$$|\varphi[\alpha \rightsquigarrow Q]| := 1 + |\varphi|$$

□

Definition 11. *The length* $|\cdot| : BP \to I\!N$ *of process expressions is given by*

$$|\alpha| := 1$$

$$|(P_1 * P_2)| := 1 + |P_1| + |P_2| \quad * \in \{;, +\}$$

□

Definition 12. *The relation* $\prec \subset BP \times HML$ *is defined by*
$(Q, \psi) \prec (P, \varphi)$ *iff* $|Q| < |P|$ *or* $(|Q| = |P|$ *and* $|\psi| < |\varphi|)$.

□

The relation $\prec$ is anti-symmetrical and transitive and $(\alpha, \top)$, $(\alpha, \bot)$ are the minimal elements.

By using $\prec$ on the set $BP \times HML$ the effect of decreasing the complexity (length) of Q by an application of the substitution operation $\varphi\{\alpha \rightsquigarrow Q\}$ is stronger than the effect of reducing the complexity of φ.

Lemma 9. *Let* $\varphi \in HML$ *be a formula and* $P \in BP$ *be a process expression. Then* $\varphi\{\alpha \rightsquigarrow P\} \in HML$ *for any* $\alpha \in \mathcal{A}$.

Proof. By well-founded induction on $\prec$.

The induction hypothesis is given by

$$\forall (Q, \psi) \prec (P, \varphi), \forall \alpha \in \mathcal{A}(\psi\{\alpha \rightsquigarrow Q\} \in HML)$$

for the process expression $P \in BP$ and the formula $\varphi \in HML$ under consideration. We consider the different cases according to Definition 9, and show for each case that the logical substitution operator strictly decreases the complexity with respect to $\prec$.

$\underline{\varphi = *, * \in \{\bot, \top\}}$:

By Definition 9 and Definition 7 we immediately conclude $\varphi\{\alpha \rightsquigarrow P\} \in HML$.

$\underline{\varphi = (\varphi_1 * \varphi_2), * \in \{\vee, \wedge\}}$:

By Definition 9 we have $(\varphi)\{\alpha \rightsquigarrow P\} = ((\varphi_1)\{\alpha \rightsquigarrow P\} * (\varphi_2)\{\alpha \rightsquigarrow P\})$. Clearly $|\varphi_i| < |\varphi|$ for $i = 1, 2$ whence by Definition 12 $(P, \varphi_i) \prec (P, \varphi)$. We get $(\varphi_1)\{\alpha \rightsquigarrow P\} = \hat{\varphi_1} \in HML$ and $(\varphi_2)\{\alpha \rightsquigarrow P\} = \hat{\varphi_2} \in HML$ by the induction hypothesis. Hence $(\hat{\varphi_1} * \hat{\varphi_2}) \in HML$.

$\varphi = [\beta]\varphi'$, $\alpha \neq \beta$:

By Definition 9 we have $(\varphi)\{\alpha \rightsquigarrow P\} = [\beta](\varphi')\{\alpha \rightsquigarrow P\}$. Now $|\varphi'| < |[\beta]\varphi'|$ whence $(P, \varphi') \prec (P, [\beta]\varphi')$ by Definition 12. Hence $(\varphi)\{\alpha \rightsquigarrow P\} = \hat{\varphi} \in HML$ and therefore $[\beta]\hat{\varphi} \in HML$.

$\varphi = [\alpha]\varphi'$, $P = \beta$:

Analogously to the above case.

$\varphi = [\alpha]\varphi'$, $P = (P_1 + P_2)$:

By Definition 9 we have

$$(\varphi)\{\alpha \rightsquigarrow P\} = ([v_n](\varphi')\{\alpha \rightsquigarrow P\})\{v_n \rightsquigarrow P_1\} \wedge ([v_m](\varphi')\{\alpha \rightsquigarrow P\})\{v_m \rightsquigarrow P_2\} \quad (1)$$

Clearly $|\varphi'| < |[\alpha]\varphi'|$ whence $(P, \varphi') \prec (P, [\alpha]\varphi')$, hence $(\varphi')\{\alpha \rightsquigarrow P\}) = \hat{\varphi} \in HML$. Hence $[v_n]\hat{\varphi} \in HML$. As $(P_1, [v_n]\hat{\varphi}) \prec (P, [\alpha]\varphi')$ since $|P_1| < |P|$ we obtain $([v_n]\hat{\varphi})\{v_n \rightsquigarrow P_1\} = \hat{\varphi_1} \in HML$. A similar argument provides $([v_m]\hat{\varphi})\{v_m \rightsquigarrow P_2\} = \hat{\varphi_2} \in HML$ for the second conjunct of (1). Hence $\hat{\varphi_1} \wedge \hat{\varphi_2} \in HML$.

$\varphi = [\alpha]\varphi'$, $P = (P_1; P_2)$:

We have $(\varphi)\{\alpha \rightsquigarrow P\} = ([v_n]([v_m](\varphi)\{\alpha \rightsquigarrow P\})\{v_m \rightsquigarrow P_2\})\{v_n \rightsquigarrow P_1\}$ by Definition 9. Since $|\varphi'| < |[\alpha]\varphi'|$ we get $(P, \varphi') \prec (P, [\alpha]\varphi')$ and therefore $(\varphi)\{\alpha \rightsquigarrow P\} = \hat{\varphi} \in HML$. Hence $[v_m]\hat{\varphi} \in HML$. Since $|P_2| < |P|$ we have $(P_2, [v_m]\hat{\varphi}) \prec (P, [\alpha]\varphi')$ and therefore $([v_m]\hat{\varphi})\{v_m \rightsquigarrow P_2\} = \hat{\varphi_1} \in HML$. By similar reasoning we conclude $([v_n]\hat{\varphi_1})\{v_n \rightsquigarrow P_2\}) = \hat{\varphi_2} \in HML$.

The proof for the diamond modality $\langle\cdot\rangle$ is similar to the cases for the box modality $[\cdot]$.

■

The following lemma can be seen as the counterpart of Remark 2 for the logical framework.

Lemma 10. *Let $Q_1, Q_2 \in BP$ be process expressions and $\varphi \in HML$ be a formula. Let $\gamma_1, \gamma_2 \in \mathcal{A}$ s.t. $\gamma_1 \neq \gamma_2$ and $* \in \{+, ;\}$. If $\gamma_1, \gamma_2 \notin \xi((Q_1 * Q_2)) \cup \xi(\varphi)$ then $(((\varphi)\{\alpha \rightsquigarrow (\gamma_1 * \gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} = (\varphi)\{\alpha \rightsquigarrow (Q_1 * Q_2)\}$.*

Proof. By induction on the structure of $\varphi \in HML$

$\varphi = *$, where $* \in \{\top, \bot\}$:

Trivial.

$\varphi = (\varphi_1 * \varphi_2)$, where $* \in \{\wedge, \vee\}$:

Follows from the induction hypothesis and Definition 9.

$\varphi = [\beta]\varphi'$, where $\alpha \neq \beta$:

Trivial, since $\gamma_1, \gamma_2 \notin \xi(\varphi)$ and no substitution takes place.

$\varphi = [\alpha]\varphi'$:

We split the proof of this step in two parts. Part one establishes the claim of the lemma for the operator ';' while part two captures the operator '+'.

Let $v_3, v_4 \notin \xi(\varphi) \cup \xi((Q_1; Q_2))$ and $v_3, v_4 \neq \gamma_i$, $i = 1, 2$

Part 1: $* =;$

We have:

$$
\begin{array}{ll}
 & ([\alpha]\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\} \\
= & ([v_3]([v_4]((\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{v_4 \rightsquigarrow \gamma_2\})\{v_3 \rightsquigarrow \gamma_1\} \\
 & \text{(By Definition 9)} \\
= & ([v_3][\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{v_3 \rightsquigarrow \gamma_1\} \\
 & \text{(By Definition 9 and } v_4 \notin \xi(\varphi'\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})) \\
= & [\gamma_1][\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\} \\
 & \text{(By Definition 9 and } v_3 \notin \xi([\gamma_2]\varphi'\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})
\end{array}
$$

Now we get

$$
\begin{array}{ll}
 & (([\gamma_1][\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \\
= & (([\gamma_1][\gamma_2]((((\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\}))\{\gamma_2 \rightsquigarrow Q_2\}) \\
 & \{\gamma_1 \rightsquigarrow Q_1\} \\
 & \text{(Since } \gamma_1, \gamma_2 \notin \xi(\varphi') \cup \xi(Q_1) \cup \xi(Q_2)) \\
= & (([\gamma_1][\gamma_2](\varphi')\{\alpha \rightsquigarrow (Q_1;Q_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \\
 & \text{(By the induction hypothesis)} \\
= & ([\gamma_1]([\gamma_2](\varphi')\{\alpha \rightsquigarrow (Q_1;Q_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \\
 & \text{(Since } \gamma_1 \neq \gamma_2) \\
= & ([\alpha]\varphi')\{\alpha \rightsquigarrow (Q_1;Q_1)\} \\
 & \text{(By Definition 13)}
\end{array}
$$

Part 2: $* = +$

We have:

$$
\begin{aligned}
& ([\alpha]\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\} \\
= \; & (([v_3](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{v_3 \rightsquigarrow \gamma_1\} \wedge ([v_4](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{v_4 \rightsquigarrow \gamma_2\}) \\
& \text{(By Definition 9)} \\
= \; & ([\gamma_1](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\} \wedge [\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\}) \\
& \text{(By Definition 9 and } v_3, v_4 \notin \xi((\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\text{)}
\end{aligned}
$$

Now we get

$$
\begin{aligned}
& ((([\gamma_1](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\} \wedge [\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\}))\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \\
= \; & (([\gamma_1](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \wedge \\
& ([\gamma_2](\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\}) \\
& \text{(By Definition 9)} \\
= \; & ((([\gamma_1]((((\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\})\{\gamma_2 \rightsquigarrow Q_2\}) \\
& \{\gamma_1 \rightsquigarrow Q_1\} \wedge \\
& (([\gamma_2]((((\varphi')\{\alpha \rightsquigarrow (\gamma_1 + \gamma_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\})\{\gamma_2 \rightsquigarrow Q_2\}) \\
& \{\gamma_1 \rightsquigarrow Q_1\}) \\
& \text{(Since } \gamma_1, \gamma_2 \notin \xi((Q_1 + Q_2)) \cup \xi(\varphi)\text{)} \\
= \; & ((([\gamma_1](\varphi')\{\alpha \rightsquigarrow (Q_1 + Q_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\} \wedge \\
& (([\gamma_2](\varphi')\{\alpha \rightsquigarrow (Q_1 + Q_2)\})\{\gamma_2 \rightsquigarrow Q_2\})\{\gamma_1 \rightsquigarrow Q_1\}) \\
& \text{(By the induction hypothesis)} \\
= \; & (([\gamma_1](\varphi')\{\alpha \rightsquigarrow (Q_1 + Q_2)\})\{\gamma_1 \rightsquigarrow Q_1\} \wedge \\
& ([\gamma_2](\varphi')\{\alpha \rightsquigarrow (Q_1 + Q_2)\})\{\gamma_2 \rightsquigarrow Q_2\}) \\
& \text{(Since } \gamma_1, \gamma_2 \notin \xi(\varphi) \cup \xi((Q_1 + Q_2)) \text{ and } \gamma_1 \neq \gamma_2\text{)} \\
= \; & ([\alpha]\varphi')\{\alpha \rightsquigarrow (Q_1 + Q_2)\} \\
& \text{(By Definition 9)}
\end{aligned}
$$

$\varphi = \langle\beta\rangle\varphi'$:

The proof is carried out in analogy to the induction step of the box-modality $[\cdot]$.
■

Definition 13. *Let $Q \in RBP$ be a process expression and $\Phi, \Phi_1, \Phi_2 \in RHML$ be formulas. We define the* logical reduction function $\mathcal{R}ed : RHML \to HML$ *as follows:*

$\mathcal{R}ed(*) := *$ *if* $* \in \{\top, \bot\}$

$\mathcal{R}ed((\Phi_1 * \Phi_2)) := (\mathcal{R}ed(\Phi_1) * \mathcal{R}ed(\Phi_2))$ *if* $* \in \{\wedge, \vee\}$

$\mathcal{R}ed([\beta]\Phi) := [\beta]\mathcal{R}ed(\Phi)$

$\mathcal{R}ed(\langle\beta\rangle\Phi) := \langle\beta\rangle\mathcal{R}ed(\Phi)$

$\mathcal{R}ed(\Phi[\alpha \rightsquigarrow Q]) := (\mathcal{R}ed(\Phi))\{\alpha \rightsquigarrow red(Q)\}$
□

Some elementary properties of the function $\mathcal{R}ed$ are given below. Lemma 11 states that one application of the reduction function is enough to remove all refinement operators occurring in a formula and is the counterpart of Remark 4 for the logical framework.

Lemma 11. *Let $Q \in RBP$ be a process expression and $\varphi \in RHML$ be a formula. Then $\mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q]) = \mathcal{R}ed(\mathcal{R}ed(\varphi)[\alpha \rightsquigarrow Q])$.*

Proof. Immediate from Definition 9 and Definition 13.
■

Lemma 12 states that the result of the reduction of formulas with nested refinements is equal to the result of the refinement on certain formulas without nested refinements. It is the 'logical' counterpart of Lemma 1.

Lemma 12. *Let $Q_1, Q_2 \in RBP$ be process expressions, $\varphi \in RHML$ be a formula and $* \in \{;, +\}$. If $\gamma_1, \gamma_2 \in \mathcal{A}$ s.t. $\gamma_1 \neq \gamma_2$ and $\gamma_1, \gamma_2 \notin \xi(\varphi) \cup \xi(red(Q_1 * Q_2))$ then $\mathcal{R}ed(((\varphi[\alpha \rightsquigarrow (\gamma_1 * \gamma_2)])[\gamma_2 \rightsquigarrow Q_2])[\gamma_1 \rightsquigarrow Q_1]) = \mathcal{R}ed(\varphi[\alpha \rightsquigarrow (Q_1 * Q_2)])$.*

Proof. Follows by Lemma 10 and Lemma 11.

The following lemma states that the logical reduction function $\mathcal{R}ed$ is always defined.

Lemma 13. *Let $\varphi \in RHML$ be a formula. Then $\mathcal{R}ed(\varphi) \in HML$.*

Proof. The proof is by induction on $|\varphi| \in \mathbb{N}$.
■

3.3 Semantics

We extend the standard satisfaction relation (see e.g. [Sti87]) to cope with refinement as follows.

Definition 14 (Satisfaction). *Let $P \in RBPP$, $Q \in RBP$ and $\varphi, \varphi_1, \varphi_2 \in RHML$.*

$P \models \top$

$P \not\models \bot$

$P \models (\varphi_1 \wedge \varphi_2)$ *iff* $P \models \varphi_1$ *and* $P \models \varphi_2$

$P \models (\varphi_1 \vee \varphi_2)$ *iff* $P \models \varphi_1$ *or* $P \models \varphi_2$

$P \models [\alpha]\varphi$ *iff* $P \in \{E \in RBPP | \forall E' \in RBPP(E \xrightarrow{\alpha} E' \Rightarrow E' \models \varphi)\}$

$P \models \langle\alpha\rangle\varphi$ *iff* $P \in \{E \in RBPP | \exists E' \in RBPP(E \xrightarrow{\alpha} E' \text{ and } E' \models \varphi)\}$

$P \models \varphi[\alpha \rightsquigarrow Q]$ *iff* $P \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$
□

We say a process P is a *model* of φ iff $P \models \varphi$. Equivalently we say P satisfies φ iff $P \models \varphi$.

Example 2. Let $P_1 := (a|b)$, $P_2 = ((a;b) + (b;a))$, $\varphi := (\langle a\rangle\langle b\rangle\top \wedge \langle b\rangle\langle a\rangle\top)$, $Q := c[c \rightsquigarrow (a_1;a_2)]$. $P_i \models \varphi$ and $P_i[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q] = (\langle a_1\rangle\langle a_2\rangle\langle b\rangle\top \wedge \langle b\rangle\langle a_1\rangle\langle a_2\rangle\top)$ for $i = 1, 2$. In addition $P_1[a \rightsquigarrow Q]$ satisfies $\langle a_1\rangle\langle b\rangle\langle a_1\rangle\top$ which is not satisfied by $P_2[a \rightsquigarrow Q]$.
□

Lemma 14. *Let $P \in RBPP$ and $\varphi \in HML$. Then $P \models \varphi \Leftrightarrow red(P) \models \varphi$.*

Proof. The proof is by induction on the structure of $\varphi \in HML$.

The base case $\varphi \in \{\top, \bot\}$ are trivial.
The cases $\varphi = (\varphi_1 * \varphi_2)$ where $* \in \{\vee, \wedge\}$ follow immediatly from Definition 14 and the induction hypothesis.

$\underline{\varphi = [\alpha]\varphi'}$:

Both directions are proved by an indirect argument.

'$\Rightarrow$':

Suppose $P \models [\alpha]\varphi'$ and $red(P) \not\models [\alpha]\varphi'$. Then

$$\exists E' \in RBPP(red(P) \xrightarrow{\alpha} E' \text{ and } E' \not\models \varphi').$$

This yields

$$\exists E' \in RBPP(P \xrightarrow{\alpha} E' \text{ and } E' \not\models \varphi')$$

whence $P \not\models [\alpha]\varphi'$.

'$\Leftarrow$':

Suppose $P \not\models [\alpha]\varphi'$ and $red(P) \models [\alpha]\varphi'$. Then

$$\exists E' \in RBPP(P \xrightarrow{\alpha} E' \text{ and } E' \not\models \varphi') \quad (1)$$

whence

$$red(P) \xrightarrow{\alpha} red(E')$$

by Lemma 8. From (1) we obtain

$$red(E') \not\models \varphi' \quad (3)$$

by the induction hypothesis. Hence (2) and (3) imply

$$\exists \tilde{E} \in RBPP(red(P) \xrightarrow{\alpha} \tilde{E} \text{ and } \tilde{E} \not\models \varphi')$$

i.e. $red(P) \not\models [\alpha]\varphi'$

$\underline{\varphi = \langle\alpha\rangle\varphi'}$:

'$\Rightarrow$': In analoy to the '$\Leftarrow$'-direction of the proof for the box modality.

'$\Leftarrow$': In analoy to the '$\Rightarrow$'-direction of the proof for the box modality.
■

Lemma 15. *Let $P \in RBPP$ and $\varphi \in RHML$. Then $P \models \varphi \Leftrightarrow P \models \mathcal{R}ed(\varphi)$.*

Proof. The proof is by induction on $|\varphi| \in \mathbb{N}$.
■

Corollary 1. *Let $P \in RBPP$ and $\varphi \in RHML$. Then $P \models \varphi \Leftrightarrow red(P) \models \varphi$.*

Proof. $P \models \varphi$

$\Leftrightarrow P \models \mathcal{R}ed(\varphi)$ (By Lemma 15)

$\Leftrightarrow red(P) \models \mathcal{R}ed(\varphi)$ (By Lemma 14 and Lemma 13)

$\Leftrightarrow red(P) \models \varphi$ (By Lemma 15)
■

4 Simultaneous Atomic Performance Refinement for *RBPP* and *RHML*

In this section we provide the link between refinement in the process calculi and the logical calculi.

4.1 The Necessary Restrictions

We give the main result (Theorem 2) in this section. It will enable us to modify both a formula $\varphi \in RHML$ and its model $P \in RBPP$ by simultaneously applying syntactic performance refinement to φ and P. The main property of the simultaneous refinement is: If process P satisfies the formula φ then the

refined process $P[a \rightsquigarrow Q]$ satisfies the refined formula $\varphi[a \rightsquigarrow Q]$ and vice versa. Formally we have: for any $P \in RBPP$, $Q \in RBP$, $\alpha \in \mathcal{A}$, $\varphi \in RHML$.

$$(P \models \varphi \Leftrightarrow P[\alpha \rightsquigarrow Q] \models \varphi[\alpha \rightsquigarrow Q]) \tag{2}$$

Assertion (2) does not hold in general but we give the (weak and reasonable) restrictions under which it is true. Actually it differs from the assertion we are interested in, namely the same assertion for the framework of action refinement. But as we will see in section 5 we obtain the desired assertion easily as a corollary of assertion (2).

It turns out that the following two conditions are sufficient to establish validity of assertion (2).

(I) $\xi(red(P)) \cap \xi(red(Q)) = \emptyset$ and

(II) $\xi(\mathcal{R}ed(\varphi)) \cap \xi(red(Q)) = \emptyset$.

We first provide a few examples to support the intuition why these conditions are necessary.
Example 3 shows a situation where a refinement does not preserve the satisfaction relation between original and refined process expressions and formulas. The situation appears due to the fact, that the original property is left unmodified but the original process is modified by the refinement.

Example 3. Consider the process expression $P := b$ and the formula $\varphi := [c]\langle d\rangle\top$.
We have $P \models \varphi$ but $red(P[b \rightsquigarrow c]) \not\models \mathcal{R}ed(\varphi[b \rightsquigarrow c])$. Note that we have $\xi(red(Q)) \cap \xi(\mathcal{R}ed(\varphi)) \neq \emptyset$.
□

We encounter another situation where the satisfaction relation is not preserved by a refinement if the refinement leaves the original process expression unmodified and modifies the original formula.

Example 4. Consider the process expression $P := b$ and the formula $\varphi := [c]\langle d\rangle\top$ from example 3.
We have $P \models \varphi$ but $red(P[c \rightsquigarrow b]) \not\models \mathcal{R}ed(\varphi[c \rightsquigarrow b])$. Note that we have $\xi(red(Q)) \cap \xi(red(P)) \neq \emptyset$.
□

The next two examples demonstrate the failure of preserving the satisfaction relation in the 'other direction', namely from the refined to the original process expressions and formulas.

Example 5. Consider the process expression $P := b$ and the formula $\varphi := \langle c\rangle\top$.
We have $P \not\models \varphi$ but $red(P[b \rightsquigarrow c]) \models \mathcal{R}ed(\varphi[b \rightsquigarrow c])$. Note that we have $\xi(red(Q)) \cap \xi(red(\varphi)) \neq \emptyset$.
□

Example 6. Consider the process expression $P := b$ and the formula $\varphi := \langle c \rangle \top$ from example 5.
We have $P \not\models \varphi$ but $red(P[c \rightsquigarrow b]) \models \mathcal{R}ed(\varphi[c \rightsquigarrow b])$. Note that we have $\xi(red(Q)) \cap \xi(red(P)) \neq \emptyset$.
□

4.2 Preliminaries for the Refinement Theorems

Two facts are crucial for the proof of the main theorem:

- We can refine a process expression $P \in RBPP$ and a formula $\varphi \in RHML$ by simple processes (which consists only of action variables) without affecting their satisfaction relation. This fact will be formalized in the 'expansion-lemmata' given below.
- We can reduce two refinement steps into one. This fact has been formalized by Lemma 1 and Lemma 12.

The following expansion lemmata formalize the possibility to refine performances of process expressions and formulas by simple process expressions, composed of two performances, without affecting the satisfaction relation.

Lemma 16 (Expansion lemma for ';'). *Let $P \in BPP$ be a process expression, $\varphi \in HML$ be a formula and $\gamma_1, \gamma_2 \in \mathcal{A}$ s.t. $\gamma_1, \gamma_2 \notin \xi(P) \cup \xi(\varphi)$. Then $P \models \varphi \Leftrightarrow red(P[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])$.*

Proof. By induction on the structure of $\varphi \in HML$

$\varphi = *$, where $* \in \{\top, \bot\}$: Trivial, since $\mathcal{R}ed(*[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) = *$.

$\varphi = (\varphi_1 * \varphi_2)$ where $* \in \{\wedge, \vee\}$:

The claim follows from the induction hypothesis and Definition 9, Definition 13 and Definition 14.

$\varphi = [\beta]\varphi'$ where $\alpha \neq \beta$:

Both directions are proved by an indirect argument.

'$\Rightarrow$':

Assume $P \models [\beta]\varphi'$ and $red(P[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \not\models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])$
From

$$red(P[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \not\models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])$$

we obtain

$$red(P[\alpha \rightsquigarrow (\gamma_1; \gamma_2)]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1; \gamma_2)])$$

by Definition 9 and Definition 13.
Hence we get

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\beta} E' \text{ and}$$

$$E' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big) \quad (1)$$

by Definition 14. Now $\alpha \neq \beta$, $\beta \notin \xi((\gamma_1;\gamma_2))$ due to the condition $\gamma_1, \gamma_2 \notin \xi(\varphi)$ and since $\beta \in \xi(\varphi)$. We obtain

$$\exists P''(P \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = E') \quad (2)$$

by application of assertion 2) from Lemma 4. Now (1) and (2) give

$$red(P''[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$$

whence we get

$$\exists P''(P \xrightarrow{\beta} P'' \text{ and } P'' \not\models \varphi')$$

by the induction hypothesis, yielding the contradiction

$$P \not\models [\beta]\varphi'.$$

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$ and $P \not\models [\beta]\varphi'$, i.e.

$$\exists E' \in RBPP(P \xrightarrow{\beta} E' \text{ and } E' \not\models \varphi')$$

Since $\alpha \neq \beta$ we obtain

$$\exists E' \in RBPP(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \text{ and } E' \not\models \varphi')$$

by application of assertion 1) from Lemma 4. But this implies

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \text{ and}$$

$$red(E'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big)$$

by the induction hypothesis. Hence we get

$$red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$$

whence the desired contradiction follows.

$\varphi = [\alpha]\varphi'$:

Claim: $\mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = [\gamma_1][\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$

Proof:

W.l.o.g. let $v_1, v_2 \notin \xi(\varphi) \cup \xi((\gamma_1;\gamma_2))$

$\mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$
$= (\mathcal{R}ed([\alpha]\varphi'))\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\}$
(By Definition 13)
$= ([\alpha]\mathcal{R}ed(\varphi'))\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\}$
(By Definition 13)
$= ([v_1]([v_2](\mathcal{R}ed(\varphi'))\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{v_2 \rightsquigarrow \gamma_2\})\{v_1 \rightsquigarrow \gamma_1\}$
(By Definition 9)
$= ([v_1][\gamma_2](\mathcal{R}ed(\varphi'))\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})\{v_1 \rightsquigarrow \gamma_1\}$
(Since $v_2 \notin \xi((\mathcal{R}ed(\varphi'))\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})$)
$= [\gamma_1][\gamma_2]\mathcal{R}ed(\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\}$
(Since $v_1 \notin \xi(\mathcal{R}ed(\varphi')\{\alpha \rightsquigarrow (\gamma_1;\gamma_2)\})$ and $v_2 \neq v_1$)
$= [\gamma_1][\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$
(By Definition 13)
□

'$\Rightarrow$':

Assume $P \models [\alpha]\varphi'$ and $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$

We have:

$red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$

$\Leftrightarrow$ $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models [\gamma_1][\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$
(By the claim)

$\Leftrightarrow$ $\exists P', P'' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'$ and $P' \xrightarrow{\gamma_2} P''$ and
$P'' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big)$ (1)

Since $\gamma_1, \gamma_2 \notin \xi(P)$ we can apply assertion 2) of Lemma 6 and obtain

$$\exists \tilde{P}(P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) = P'') \quad (2)$$

Taking (1) and (2) together we have $red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$. By the induction hypothesis we obtain

$$\tilde{P} \not\models \varphi' \quad (3)$$

But (2) and (3) imply

$$P \not\models [\alpha]\varphi'.$$

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$ and $P \not\models [\alpha]\varphi'$. From the latter we obtain

$$\exists P' \in RBPP(P \xrightarrow{\alpha} P' \text{ and } P' \not\models \varphi').$$

By assertion 1) of Lemma 6 we get

$$\exists P''(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} P'' \xrightarrow{\gamma_2} red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])) \quad (1)$$

Further $P' \not\models \varphi'$ implies

$$red(P'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \quad (2)$$

by the induction hypothesis. By (1) and (2)

$$\exists E', E'' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\gamma_1} E' \xrightarrow{\gamma_2} E'' \text{ and}$$

$$E'' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])\Big)$$

and therefore

$$red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]).$$

$\varphi = \langle\beta\rangle\varphi'$ where $\alpha \neq \beta$:

Here we use a direct argument to provide the desired result. However we only indicate the proofs since they have their analogous counterparts in the proofs for the box-modality $[\cdot]$.

'$\Rightarrow$':

Assume $P \models \langle\beta\rangle\varphi'$. Then

$$\exists E' \in RBPP(P \xrightarrow{\beta} E' \text{ and } E' \models \varphi').$$

It is easy to see that the proof can now be carried out in analogy to the proof of the '$\Leftarrow$'-direction in the induction step for $\varphi = [\beta]\varphi'$ where $\alpha \neq \beta$ whence $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \models \mathcal{R}ed((\langle\beta\rangle\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$.

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \models \mathcal{R}ed((\langle\beta\rangle\varphi')[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$, hence

$$red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \models \langle\beta\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])$$

whence

$$\exists E' \in RBPP(red(P[\alpha \rightsquigarrow (\gamma_1;\gamma_2)]) \xrightarrow{\beta} E' \text{ and } E' \models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1;\gamma_2)])).$$

We can now proceed in analogy to the proof of the '$\Rightarrow$'-direction in the induction step for $\varphi = [\beta]\varphi'$ where $\alpha \neq \beta$ whence $P \models \langle\beta\rangle\varphi'$.

$\underline{\varphi = \langle\alpha\rangle\varphi'}$:

'$\Rightarrow$': In analogy to the '$\Leftarrow$'-direction of the induction step for $\varphi = [\alpha]\varphi'$.

'$\Leftarrow$': In analogy to the '$\Rightarrow$'-direction of the induction step for $\varphi = [\alpha]\varphi'$.
■

Lemma 17 (Expansion lemma for '+'). *Let $P \in BPP$ be a process expression, $\varphi \in HML$ be a formula and $\gamma_1, \gamma_2 \in \mathcal{A}$ be s.t. $\gamma_1, \gamma_2 \notin \xi(P) \cup \xi(\varphi)$. Then $P \models \varphi \Leftrightarrow red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$.*

Proof. The proof is by induction on the structure of $\varphi \in BPP$. Most of the induction steps are carried out in analogy to those steps in the proof of Lemma 16. However two cases are different and will be shown below.

$\underline{\varphi \in \{\top, \bot\}}$: Trivial.

$\underline{\varphi = (\varphi_1 * \varphi_2) \text{ where } * \in \{\vee, \wedge\}}$: In analogy to the proof of Lemma 16.

$\underline{\varphi = [\beta]\varphi' \text{ where } \beta \neq \alpha}$: In analogy to the proof of Lemma 16.

$\underline{\varphi = \langle\beta\rangle\varphi' \text{ where } \beta \neq \alpha}$: In analogy to the proof of Lemma 16.

$\underline{\varphi = [\alpha]\varphi'}$:

Both directions are shown by an indirect argument.

Claim: $\mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) =$

$$([\gamma_1]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \wedge [\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

This can easily be shown by Definition 9 and Definition 13.
□

'$\Rightarrow$':

Assume $P \models [\alpha]\varphi'$ and $red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$.

By the assumption and the claim we obtain

$red(P[\alpha \rightsquigarrow (\gamma_1+\gamma_2)]) \not\models ([\gamma_1]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1+\gamma_2)])\wedge[\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1+\gamma_2)]))$

which is equivalent to

$$\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models [\gamma_1]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

or

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models [\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big)$$

W.l.o.g. assume the former. Then we have

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_1} E' \text{ and}$$

$$E' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big) \quad (1)$$

By the condition of the lemma we have $\gamma_1, \gamma_2 \notin \xi(P)$ whence

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) = E') \quad (2)$$

by the application of assertion 2) from Lemma 7. Now (1) and (2) together give

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

whence we obtain

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } \tilde{P} \not\models \varphi')$$

by the induction hypothesis, i.e.

$$P \not\models [\alpha]\varphi'.$$

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$ and $P \not\models [\alpha]\varphi'$. From the latter we obtain

$$\exists E' \in RBPP(P \xrightarrow{\alpha} E' \text{ and } E' \not\models \varphi').$$

By assertion 1) of Lemma 7

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \text{ and}$$

$$E' \not\models \varphi' \Big) \text{ for } i = 1, 2$$

and therefore we get

$$\exists E' \in RBPP \Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \text{ and}$$

$$red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \Big) \text{ for } i = 1, 2$$

by the induction hypothesis. Hence

$$\exists E' \in RBPP \Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} E' \text{ and}$$

$$E' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \Big) \text{ for } i = 1, 2$$

which shows

$$\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models [\gamma_1]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

and

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models [\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \Big)$$

by Definition 14.

Hence we obtain

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models$$
$$([\gamma_1]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \wedge [\gamma_2]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

and therefore

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

by the claim.

$\underline{\varphi = \langle\alpha\rangle\varphi'}$:

Claim: $\mathcal{R}ed((\langle\alpha\rangle\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) =$

$$(\langle\gamma_1\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \wedge \langle\gamma_2\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

This can easily be shown by Definition 9 and Definition 13.
□

'$\Rightarrow$':

Assume $P \models \langle\alpha\rangle\varphi'$. Then

$$\exists E' \in RBPP(P \xrightarrow{\alpha} E' \text{ and } E' \models \varphi').$$

By assertion 1) of Lemma 7

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \text{ and}$$

$$E' \models \varphi'\Big) \text{ for } i = 1, 2$$

and therefore we get

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \text{ and}$$

$$red(E'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big) \text{ for } i = 1, 2$$

by application of the induction hypothesis. Hence

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} E' \text{ and}$$

$$E' \models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big) \text{ for } i = 1, 2$$

which shows

$$\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \langle\gamma_1\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

and

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \langle\gamma_2\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big).$$

Hence we obtain

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models$$
$$(\langle\gamma_1\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \wedge \langle\gamma_2\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

and therefore

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed((\langle\alpha\rangle\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

by the claim.

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed((\langle\alpha\rangle\varphi')[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$.

By the assumption and the claim we obtain

$$red(P[\alpha \rightsquigarrow (\gamma_1+\gamma_2)]) \models (\langle\gamma_1\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1+\gamma_2)])\wedge\langle\gamma_2\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1+\gamma_2)]))$$

which is equivalent to

$$\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \langle\gamma_1\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])$$

and

$$red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \langle\gamma_2\rangle\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big)$$

Then we have

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \xrightarrow{\gamma_i} E' \text{ and}$$

$$E' \models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)])\Big) \text{ for } i = 1, 2 \quad (1)$$

By the condition of the lemma we have $\gamma_1, \gamma_2 \notin \xi(P)$ whence

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) = E') \quad (2)$$

by the application of assertion 2) from Lemma 7. Now (1) and (2) together give

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } red(\tilde{P}[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]) \models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (\gamma_1 + \gamma_2)]))$$

whence we obtain

$$\exists \tilde{P} \in RBPP(P \xrightarrow{\alpha} \tilde{P} \text{ and } \tilde{P} \models \varphi')$$

by the application of the induction hypothesis. Hence we conclude

$$P \models \langle\alpha\rangle\varphi'$$

by Definition 14 which completes this case

■

4.3 The Refinement Theorem for Simultaneous Performance Refinement

In this section we provide the link between syntactic action refinement for the languages $RBPP$ and $RHML$.

Theorem 1. *Let $P \in BPP$ and $Q \in RBP$. Further let $\varphi \in HML$. If $\xi(P) \cap \xi(red(Q)) = \emptyset$ and $\xi(\varphi) \cap \xi(red(Q)) = \emptyset$ then $P \models \varphi \Leftrightarrow red(P[\alpha \rightsquigarrow Q]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$.*

Proof. By induction on the structure of $\varphi \in HML$ and a subsidiary induction on the structure of $Q \in RBP$.

$\varphi = *$, where $* \in \{\top, \bot\}$: Trivial.

$\varphi = (\varphi_1 * \varphi_2)$ where $* \in \{\wedge, \vee\}$: Routine. $(\star)$

$\varphi = [\beta]\varphi'$ where $\alpha \neq \beta$:

Both directions are proved by an indirect argument.

'$\Rightarrow$':

In this case we use the condition $\xi(\varphi) \cap \xi(red(Q))$ imposed on the theorem.

Assume $P \models [\beta]\varphi'$ and $red(P[\alpha \rightsquigarrow Q]) \not\models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow Q])$
From

$$red(P[\alpha \rightsquigarrow Q]) \not\models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow Q])$$

we obtain

$$red(P[\alpha \rightsquigarrow Q]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow Q]).$$

Hence

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} E' \text{ and}$$

$$E' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow Q])\Big) \quad (1)$$

by Definition 14. Now $\alpha \neq \beta$ by the current induction step. Further $\beta \notin \xi(red(Q))$ due to the condition of the theorem and since $\beta \in \xi(\varphi)$. Hence

$$\exists P''(P \xrightarrow{\beta} P'' \text{ and } red(P''[\alpha \rightsquigarrow Q]) = E') \quad (2)$$

by application of assertion 2) from Lemma 4. Now (1) and (2) give

$$red(P''[\alpha \rightsquigarrow Q]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow Q])$$

whence we obtain

$$\exists P''(P \xrightarrow{\beta} P'' \text{ and } P'' \not\models \varphi')$$

by the induction hypothesis, i.e.

$$P \not\models [\beta]\varphi'.$$

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow Q]) \models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow Q])$ and $P \not\models [\beta]\varphi'$. From $P \not\models [\beta]\varphi'$ we obtain

$$\exists E' \in RBPP(P \xrightarrow{\beta} E' \text{ and } E' \not\models \varphi').$$

Since $\alpha \neq \beta$ by the current induction step we obtain

$$\exists E' \in RBPP(red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow Q]) \text{ and } E' \not\models \varphi')$$

by application of assertion 1) from Lemma 4. But this implies

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow Q]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow Q]) \text{ and}$$

$$red(E'[\alpha \rightsquigarrow Q]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow Q])\Big)$$

by the contrapositiv application of the induction hypothesis. Hence

$$red(P[\alpha \rightsquigarrow Q]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow Q])$$

i.e.

$$red(P[\alpha \rightsquigarrow Q]) \not\models \mathcal{R}ed(([\beta]\varphi')[\alpha \rightsquigarrow Q]).$$

$\varphi = [\alpha]\varphi'$:

(Subinduction on the structure of $Q \in RBP$)

a) $Q = \beta$:

Both directions are proved by means of an indirect argument.

'$\Rightarrow$':

To prove this direction we use the condition $\xi(P) \cap \xi(red(Q))$ imposed on the theorem.

Assume $P \models [\alpha]\varphi'$ and $red(P[\alpha \rightsquigarrow \beta]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow \beta])$
From

$$red(P[\alpha \rightsquigarrow \beta]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow \beta])$$

we obtain

$$red(P[\alpha \rightsquigarrow \beta]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow \beta]).$$

Hence

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} E' \text{ and}$$

$$E' \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow \beta])\Big) \quad (1)$$

Now $\beta \in \xi(red(Q))$ implies $\beta \notin \xi(P)$ whence

$$\exists P''(P \xrightarrow{\alpha} P'' \text{ and } red(P''[\alpha \rightsquigarrow \beta]) = E') \quad (2)$$

by application of assertion 2) from Lemma 5. Now (1) and (2) give

$$red(P''[\alpha \rightsquigarrow \beta]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow \beta])$$

whence

$$\exists P''(P \xrightarrow{\alpha} P'' \text{ and } P'' \not\models \varphi')$$

by the induction hypothesis, i.e.

$$P \not\models [\alpha]\varphi'$$

'$\Leftarrow$':

Assume $red(P[\alpha \rightsquigarrow \beta]) \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow Q])$ and $P \not\models [\alpha]\varphi'$. From $P \not\models [\alpha]\varphi'$ we obtain

$$\exists E' \in RBPP(P \xrightarrow{\alpha} E' \text{ and } E' \not\models \varphi').$$

Hence

$$\exists E' \in RBPP(red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow \beta]) \text{ and } E' \not\models \varphi')$$

by application of assertion 1) from Lemma 5. But this implies

$$\exists E' \in RBPP\Big(red(P[\alpha \rightsquigarrow \beta]) \xrightarrow{\beta} red(E'[\alpha \rightsquigarrow \beta]) \text{ and}$$

$$red(E'[\alpha \rightsquigarrow \beta]) \not\models \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow \beta])\Big)$$

by the induction hypothesis. Hence

$$red(P[\alpha \rightsquigarrow \beta]) \not\models [\beta]\mathcal{R}ed(\varphi'[\alpha \rightsquigarrow \beta])$$

i.e.

$$red(P[\alpha \rightsquigarrow \beta]) \not\models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow \beta])$$

b) $Q = (Q_1 + Q_2)$:

W.l.o.g. let $v_1, v_2 \notin \xi(P) \cup \xi(red(Q)) \cup \xi(\varphi)$.

Let $\hat{P} := red(P[\alpha \rightsquigarrow (v_1 + v_2)]$ and $\hat{\varphi} := \mathcal{R}ed(\varphi'[\alpha \rightsquigarrow (v_1 + v_2)])$.

We have $P \models \varphi$

$\Leftrightarrow$ $\hat{P} \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (v_1 + v_2)])$
(By Lemma 17)

$\Leftrightarrow$ $\hat{P} \models ([v_1]\hat{\varphi} \wedge [v_2]\hat{\varphi})$
(By the claim used in Lemma 17)

$\Leftrightarrow$ $red(\hat{P}[v_2 \rightsquigarrow Q_2]) \models \mathcal{R}ed((([v_1]\hat{\varphi} \wedge [v_2]\hat{\varphi}))[v_2 \rightsquigarrow Q_2])$
(By the induction step $(\star)$)

$\Leftrightarrow$ $red(\hat{P}[v_2 \rightsquigarrow Q_2]) \models ((\mathcal{R}ed([v_1]\hat{\varphi}))\{v_2 \rightsquigarrow Q_2\} \wedge (\mathcal{R}ed([v_2]\hat{\varphi}))\{v_2 \rightsquigarrow Q_2\})$
(By Definition 9 and Definition 13)

$\Leftrightarrow$ $red((red(\hat{P}[v_2 \rightsquigarrow Q_2]))[v_1 \rightsquigarrow Q_1]) \models$
$\mathcal{R}ed((((\mathcal{R}ed([v_1]\hat{\varphi}))\{v_2 \rightsquigarrow Q_2\} \wedge (\mathcal{R}ed([v_2]\hat{\varphi}))\{v_2 \rightsquigarrow Q_2\}))[v_1 \rightsquigarrow Q_1])$
(By induction)

$\Leftrightarrow$ $red((red(\hat{P}[v_2 \rightsquigarrow Q_2]))[v_1 \rightsquigarrow Q_1]) \models$
$\mathcal{R}ed((\mathcal{R}ed((\mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (v_1 + v_2)]))[v_2 \rightsquigarrow Q_2]))[v_1 \rightsquigarrow Q_1])$
(By Definition 9 and Definition 13)

$red(((P[\alpha \rightsquigarrow (v_1 + v_2)])[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1]) \models$
$\mathcal{R}ed(((\varphi[\alpha \rightsquigarrow (v_1 + v_2)])[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1])$
(By Remark 4 and Lemma 11)

$\Leftrightarrow$ $red(P[\alpha \rightsquigarrow (Q_1 + Q_2)]) \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (Q_1 + Q_2)])$
(By Lemma 1 and Lemma 12)

c) $Q = (Q_1; Q_2)$:

W.l.o.g. let $v_1, v_2 \notin \xi(P) \cup \xi(red(Q)) \cup \xi(\varphi)$

Let $\hat{P} := red(P[\alpha \rightsquigarrow (v_1; v_2)]$ and $\hat{\varphi} := \mathcal{R}ed(\varphi[\alpha \rightsquigarrow (v_1; v_2)])$.

We have $P \models \varphi$

$\Leftrightarrow$ $\hat{P} \models \hat{\varphi}$
(By Lemma 16)

$\Leftrightarrow$ $red(red(\hat{P}[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1]) \models \mathcal{R}ed(\mathcal{R}ed(\hat{\varphi}[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1])$
(By induction)

$\Leftrightarrow$ $red(((P[\alpha \rightsquigarrow (v_1; v_2)])[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1]) \models$
$\mathcal{R}ed(((\varphi[\alpha \rightsquigarrow (v_1; v_2)])[v_2 \rightsquigarrow Q_2])[v_1 \rightsquigarrow Q_1])$
(By Remark 4 and Lemma 11)

$\Leftrightarrow$ $red(P[\alpha \rightsquigarrow (Q_1;Q_2)]) \models \mathcal{R}ed(([\alpha]\varphi')[\alpha \rightsquigarrow (Q_1;Q_2)])$
(By Lemma 1 and Lemma 12)

d) $Q = Q_1[\gamma \rightsquigarrow Q_2]$ where $Q_1, Q_2 \in RBP$ and $\gamma \in \mathcal{A}$:

We have $red(P[\alpha \rightsquigarrow Q]) = red(P[\alpha \rightsquigarrow red(Q)])$ by Definition 3 and Definition 4. Further we have $\mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q]) = \mathcal{R}ed(\varphi[\alpha \rightsquigarrow red(Q)])$ by Definition 9 and Definition 13. Then

$$red(P[\alpha \rightsquigarrow Q]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$$

iff

$$red(P[\alpha \rightsquigarrow red(Q)]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow red(Q)])$$

iff

$$P \models \varphi$$

The last equivalence holds since $red(Q) \in BP$ thereby reducing the current induction step to one of the previous steps in the induction on the structure of $Q \in RBP$.

$\varphi = \langle\beta\rangle\varphi'$ where $\beta \neq \alpha$:

'$\Rightarrow$': The proof of this direction proceeds in analogy to the '$\Leftarrow$'-direction of the box modality $[\cdot]$.

'$\Leftarrow$': The proof of this direction proceeds in analogy to the '$\Rightarrow$'-direction of the box modality $[\cdot]$.

$\varphi = \langle\alpha\rangle\varphi'$: For the subsidiary induction step $Q = \beta$ the proof of the '$\Rightarrow$'-direction proceeds in analogy to the '$\Leftarrow$'-direction of this induction step for the box modality $[\cdot]$ and vive versa. The other steps of the induction on the structure of $Q \in RBP$ are similar to those steps for the box modality.
■

Theorem 2 (Simultaneous Performance Refinement Theorem). *Let $P \in RBPP$, $Q \in RBP$, $\alpha \in \mathcal{A}$ and $\varphi \in RHML$. If $\xi(red(P)) \cap \xi(red(Q)) = \emptyset$ and $\xi(\mathcal{R}ed(\varphi)) \cap \xi(red(Q)) = \emptyset$ then $P \models \varphi \Leftrightarrow P[\alpha \rightsquigarrow Q] \models \varphi[\alpha \rightsquigarrow Q]$.*

Proof. We have $P \models \varphi$

iff $red(P) \models \varphi$ (By Corollary 1)

iff $red(P) \models \mathcal{R}ed(\varphi)$ (By Lemma 15)

iff $red(red(P)[\alpha \rightsquigarrow Q]) \models \mathcal{R}ed(\mathcal{R}ed(\varphi)[\alpha \rightsquigarrow Q])$ (By Theorem 1)

iff $red(P[\alpha \rightsquigarrow Q]) \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$ (By Remark 4 and Lemma 11)

iff $P[\alpha \rightsquigarrow Q] \models \mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$ (By Corollary 1)

iff $P[\alpha \rightsquigarrow Q] \models \varphi[\alpha \rightsquigarrow Q]$ (By Lemma 15)
■

5 Simultaneous Action Refinement for $RBPP$ and $RHML$

In this section we give restate Theorem 2 of the previous section in the environment of action refinement. Therefore we consider the set Act instead of the set $\mathcal{A}$ for the provision of conceptual entities:

> We consider the process language $RBPP$ of section 2 (Definition 1) where the rule $PE ::= \alpha$ is replaced by the rule $PE ::= a$ and $a \in Act$. Call the language generated by this grammar $RBPP_{pure}$. Clearly Theorem 2 of section 4 still holds if we replace $RBPP$ (RBP) by $RBPP_{pure}$ (RBP_{pure} resp.) since $RBPP_{pure} \subseteq RBPP$ ($RBP_{pure} \subseteq RBP$). The same claim holds for $RHML_{pure}$ which is the language generated by the language $RHML$ (Definition 7 in section 3) where the rules $\Phi ::= [\alpha]\Phi$ and $\Phi ::= \langle\alpha\rangle\Phi$ are replaced by the rules $\Phi ::= [a]\Phi$ and $\Phi ::= \langle a\rangle\Phi$ where $a \in Act$.

The following is immediate.

Theorem 3 (Simultaneous Action Refinement Theorem).
Let $P \in RBPP_{pure}$, $Q \in RBP_{pure}$ be process expressions, $\varphi \in RHML_{pure}$ be a formula and $a \in Act$ be an atomic action. If $\xi_{Act}(red(P)) \cap \xi_{Act}(red(Q)) = \emptyset$ and $\xi_{Act}(\mathcal{R}ed(\varphi)) \cap \xi_{Act}(red(Q)) = \emptyset$ then $P \models \varphi \Leftrightarrow P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$.

Example 7. We take $\varphi = [a](\langle b\rangle\top \vee [a]\bot)$ and $Q = (d + e)$ where $a, b, d, e \in \mathcal{A}$ from Example 1. Further let $P_1 = (a; (b + a))$ and $P_2 = ((a; b) + (a; a))$. Now $P_1 \models \varphi$ which by Theorem 2 implies $P_1[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$. On the other hand we have $P_2 \not\models \varphi$ whence by Theorem 2 we obtain $P_2[a \rightsquigarrow Q] \not\models \varphi[a \rightsquigarrow Q]$.
□

6 Conclusion

We defined syntactic performance refinement on a subset of CCS-process expressions and formulas of the Hennessy-Milner-Logic. We proved that for $P \in RBPP_{pure}$, $Q \in RBP_{pure}$, $a \in Act$, $\varphi \in RHML_{pure}$ the assertion

$$P \models \varphi \Leftrightarrow P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$$

holds under the two liberal restrictions $\xi_{Act}(red(P)) \cap \xi_{Act}(red(Q)) = \emptyset$ and $\xi_{Act}(\mathcal{R}ed(\varphi)) \cap \xi_{Act}(red(Q)) = \emptyset$ which can be dropped if we rename the actions of $red(Q)$ in the obvious way. Hence a system designer can simultaneously refine the syntax of a process expression and a formula in a stepwise manner (using Theorem 3) as illustrated in Figure 3 while preserving the satisfaction relation.

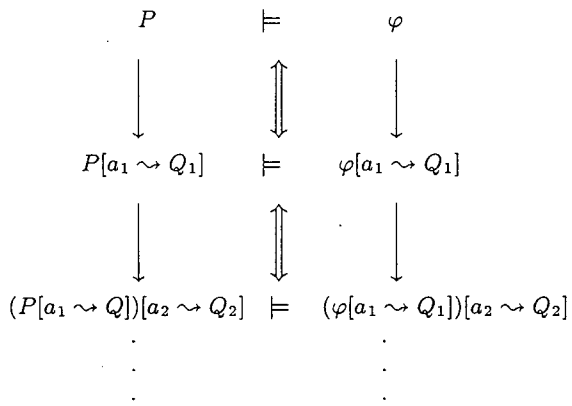

Fig. 3. Stepwise Simultaneous Action Refinement

Another interesting result is that the assertion[1]

$$(P_1 \equiv_b P_2) \Rightarrow \Big(P_1 \models \varphi \Rightarrow (P_2[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q])\Big)$$

holds for any $P_1, P_2 \in BPP_{pure}$ due to the following fact: The assertion $P_1 \equiv_b P_2$ implies the assertion $\forall \varphi \in HML_{pure}.(P_1 \models \varphi \Leftrightarrow P_2 \models \varphi)$ (cf. [Mil80,Sti96] for the general result). It is folklore that $P_1 \equiv_b P_2$ not necessarily implies $red(P_1[a \rightsquigarrow Q]) \equiv_b red(P_2[a \rightsquigarrow Q])$. In our framework, a system designer can first establish the assertion $P \models \varphi$. Then she applies simultaneous action refinement obtaining $\hat{P} \models \hat{\varphi}$ where

$$\hat{P} = (\ldots(P[a_1 \rightsquigarrow Q_1])[a_2 \rightsquigarrow Q_2])\ldots)[a_n \rightsquigarrow Q_n]$$

and

$$\hat{\varphi} = (\ldots(\varphi[a_1 \rightsquigarrow Q_1])[a_2 \rightsquigarrow Q_2])\ldots)[a_n \rightsquigarrow Q_n]$$

Replacing the expression $E \in BPP_{pure}$ for P where $E \equiv_b P$ a system designer immediatly knows $\hat{E} \models \hat{\varphi}$ where

$$\hat{E} = (\ldots(E[a_1 \rightsquigarrow Q_1])[a_2 \rightsquigarrow Q_2])\ldots)[a_n \rightsquigarrow Q_n]$$

[1] $\equiv_b$ is the denotation of strong bisimulation equivalence defined in [Mil80].

The generalisation of this result to sets of properties is straightforward and sheds a different light on the statement that interleaving semantics are not appropriate to model action refinement: A designer might not be interested in full equality of processes modulo an equivalence relation, but in partial equality modulo a defined set of properties. This fact can be exploited to circumvent the problems which occur when action refinement operators are used in a framework where concurrency is modelled by interleaving of actions.

Verification of programs is commonly formalized by means of implementation relations (cf. [BBR90]). Since atomic actions are uninterpreted 'action names', there is no such relation between an action a and an expression Q in the term $P[a \rightsquigarrow Q]$. Although our approach is not appropriate for this kind of verification it supports 'a priory'-verification with respect to transition behaviour formalized by formulas of the language $RHML$.

Work is in progress that extends the above results: We model infinite behaviour, study synchronisation, and investigate other logics and appropriate models for concurrency. Furthermore we investigate polynomial time algorithms for the implementation of the reduction functions red and $\mathcal{R}ed$.

A future topic is the investigation of the equivalence relations induced on the process environments by the extended logical frameworks.

References

[AH91] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Lecture Notes in Computer Science*, 510:506–519, 1991.

[BBR90] In W. P. De Roever G. Rozenberg J. W. De Bakker, editor, REX Workshop on *Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness*, Mook, The Netherlands, May/June 1989, volume 430 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BDE93] E. Best, R. Devillers, and J. Esparza. General refinement and recursion operators for the Petri box calculus. *Lecture Notes in Computer Science*, 665:130–140, 1993.

[DD93] Ph. Darondeau and P. Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 118(1):21–48, September 1993.

[Eme90] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072, Amsterdam, 1990. Elsevier Science Publishers.

[GGR94] U. Goltz, R. Gorrieri, and A. Rensink. On syntactic and semantic action refinement. *Lecture Notes in Computer Science*, 789:385–404, 1994.

[Hoa90] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, Englewood Cliffs, 1990.

[Huh96] Michaela Huhn. Action refinement and property inheritance in systems of sequential agents. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *Lecture Notes in Computer Science*, pages 639–654, Pisa, Italy, 26–29 August 1996. Springer-Verlag.

[Koz83] D. Kozen. Results on the propositional mu -calculus. *Theoretical Computer Science*, 27(3):333–354, December 1983.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Springer, Berlin, 1 edition, 1980.

[Pen91] W. Penczek. Branching time and partial order in temporal logic. Technical Report UMCS–91–3–3, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, 1991.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49(2-3):311–347, 1987.

[Sti96] C. Stirling. Modal and temporal logics for processes. *Lecture Notes in Computer Science*, 1043:149–237, 1996.

[vGG89] R. van Glabbeek and U. Goltz. Partial order semantics for refinement of actions–neither necessary nor always sufficient but appropriate when used with care–. *BEATCS: Bulletin of the European Association for Theoretical Computer Science*, 38, 1989.