Decompositional Reasoning about the History of Parallel Processes*

Luca Aceto[1], Arnar Birgisson[1,2],
Anna Ingolfsdottir[1], and MohammadReza Mousavi[3]

[1] School of Computer Science, Reykjavik University, Iceland
[2] Department of Computer Science and Engineering,
Chalmers University of Technology, Sweden
[3] Department of Computer Science, TU/Eindhoven, The Netherlands

Abstract. Decompositional reasoning aims at automatically decomposing a global property of a composite system into local properties of (possibly unknown) components. In concurrency theory, decompositional reasoning techniques date back to the seminal work of Larsen and Liu in the late 1980s and early 1990s. However, we are not aware of any such decomposition technique that applies to reasoning about the "past". In this paper, we address this problem and present a decomposition technique for Hennessy-Milner logic with past and its extension with recursively defined formulae. As a language for processes, we use a subset of Milner's CCS with parallel composition, non-deterministic choice, action prefixing and the inaction constant. We focus on developing decompositional reasoning techniques for parallel contexts in that language.

1 Introduction

State-space explosion is a major obstacle in model checking logical properties. One approach to combat this problem is compositional reasoning, where properties of a system as a whole are deduced in a principled fashion from properties of its components. The study of compositional proof systems for various temporal and modal logics has attracted considerable attention in the concurrency-theory literature and several compositional proof systems have been proposed for such logics over (fragments of) process calculi. (See, e.g., [3, 23, 24, 27].) A related line of research is the one devoted to (de)compositional model checking [2, 9, 15, 20, 28]. Decompositional reasoning aims at automatically decomposing the global property to be model checked into local properties of (possibly unknown) components—a technique that is often called *quotienting*. In the context of process algebras, as the language for describing reactive systems, and (extensions

* The work of Aceto, Birgisson and Ingolfsdottir has been partially supported by the project "New Developments in Operational Semantics" (nr. 080039021) of the Icelandic Research Fund. Birgisson has been further supported by research-student grant nr. 080890008 of the Icelandic Research Fund and by grants from the Swedish research agencies SSF and VR.

of) Hennessy-Milner logic (HML), as the logical specification formalism for describing their properties, decompositional reasoning techniques date back to the seminal work of Larsen and Liu in the 1980's and early 1990's [18, 20], which is further developed in, e.g., [4, 8, 13]. However, we are not aware of any such decomposition technique that applies to reasoning about the "past". This is particularly interesting in the light of recent developments concerning reversible processes [22] and knowledge representation (epistemic aspects) inside process algebra [6, 10], all of which involve some notion of specification and reasoning about the past. Moreover, a significant body of evidence indicates that being able to reason about the past is useful in program verification [12, 17, 21].

In this paper, we address the problem of developing a decomposition technique for Hennessy-Milner logic with past and for its extension with recursively defined formulae. As the language for describing processes, in order to highlight the main ideas and technical tools in our approach, we use a subset of CCS with parallel composition, non-deterministic choice, action prefixing and the inaction constant. As the work presented in this paper shows, the development of a theory of decompositional reasoning in a setting with past modalities involves subtleties and design decisions that do not arise in previous work on HML and Kozen's μ-calculus [14].

The rest of this paper is structured as follows. Section 2 introduces preliminary definitions and the extension of Hennessy-Milner logic with past. Section 3 discusses how parallel computations are decomposed into their components. Section 4 presents the decompositional reasoning technique and the main theorem of the paper. Section 5 extends the theory to recursively defined formulae, and Section 6 discusses related work and possible extensions of our results.

2 Preliminaries

Definition 1 (Labelled transition system). *A* labelled transition system *(LTS) is a triple* $\langle P, A, \longrightarrow \rangle$ *where*

- P *is a set of process names,*
- A *is a finite set of action names, not including a* silent action τ *(we write* A_τ *for* $A \cup \{\tau\}$*), and*
- $\longrightarrow \subseteq P \times A_\tau \times P$ *is the* transition relation*; we call its elements* transitions *and usually write* $p \xrightarrow{\alpha} p'$ *to mean that* $(p, \alpha, p') \in \longrightarrow$.

We let $p, q, \ldots$ *range over* P*,* $a, b, \ldots$ *over* A *and* $\alpha, \beta, \ldots$ *over* A_τ.

Definition 2 (Sequences and computations). *For any set* S*, we let* S^* *be the set of finite sequences of elements from* S*. Concatenation of sequences is represented by juxtaposition.* λ *denotes the empty sequence and* $|w|$ *stands for the length of a sequence* w.

Given an LTS $\mathcal{T} = \langle P, A, \longrightarrow \rangle$*, we define a* path from p_0 *to be a sequence of transitions* $p_0 \xrightarrow{\alpha_0} p_1$, $p_1 \xrightarrow{\alpha_1} p_2$, $\ldots$, $p_{n-1} \xrightarrow{\alpha_{n-1}} p_n$ *and usually write this as* $p_0 \xrightarrow{\alpha_0} p_1 \xrightarrow{\alpha_1} p_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_{n-1}} p_n$.

We use $\pi, \mu, \ldots$ to range over paths. A computation from p *is a pair* (p, π) *where* π *is a path from* p *and we use* $\rho, \rho', \ldots$ *to range over computations.* $\mathcal{C}_{\mathcal{T}}(p)$*, or simply* $\mathcal{C}(p)$ *when the LTS* $\mathcal{T}$ *is clear from the context, is the set of computations from* p *and* $\mathcal{C}_{\mathcal{T}}$ *is the set of all computations in* $\mathcal{T}$*.*

For a computation $\rho = (p_0, \pi)$ *where* $\pi = p_0 \xrightarrow{\alpha_0} p_1 \xrightarrow{\alpha_1} p_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_{n-1}} p_n$ *we define* $\mathrm{first}(\rho) = \mathrm{first}(\pi) = p_0$, $\mathrm{last}(\rho) = \mathrm{last}(\pi) = p_n$, $\mathrm{trace}(\rho) = (\alpha_0 \ldots \alpha_{n-1}) \in A_\tau^*$ *and* $|\rho| = |\pi| = n$*. We refer to the elements of* A_τ^* *as* traces*.*

Concatenation of computations ρ *and* ρ' *is denoted by their juxtaposition* $\rho\rho'$ *and is defined iff* $\mathrm{last}(\rho) = \mathrm{first}(\rho')$*. When* $\mathrm{last}(\rho) = p$ *we write* $\rho(p \xrightarrow{\alpha} q)$ *as a shorthand for the slightly longer* $\rho(p, p \xrightarrow{\alpha} q)$*. We also use* $\rho \xrightarrow{\alpha} \rho'$ *to denote that there exists a computation* $\rho'' = (p, p \xrightarrow{\alpha} p')$*, for some processes* p *and* p'*, such that* $\rho' = \rho\rho''$*.*

Definition 3 (Hennessy-Milner logic with past). *Let* $\mathcal{T} = \langle P, A, \rightarrow \rangle$ *be an LTS. The set* $HML_{\leftarrow}(A)$*, or simply* $HML_{\leftarrow}$*, of* Hennessy-Milner logic formulae with past *is defined by the following grammar, where* $\alpha \in A_\tau$*.*

$$\varphi, \psi ::= \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \langle{\leftarrow}\alpha\rangle\varphi.$$

We define the satisfaction relation $\models \subseteq \mathcal{C}_{\mathcal{T}} \times HML_{\leftarrow}$ *as the least relation that satisfies the following clauses:*

- $\rho \models \top$ *for all* $\rho \in \mathcal{C}_{\mathcal{T}}$*,*
- $\rho \models \varphi \wedge \psi$ *iff* $\rho \models \varphi$ *and* $\rho \models \psi$*,*
- $\rho \models \neg\varphi$ *iff not* $\rho \models \varphi$*,*
- $\rho \models \langle\alpha\rangle\varphi$ *iff* $\rho \xrightarrow{\alpha} \rho'$ *and* $\rho' \models \varphi$ *for some* $\rho' \in \mathcal{C}_{\mathcal{T}}$*, and*
- $\rho \models \langle{\leftarrow}\alpha\rangle\varphi$ *iff* $\rho' \xrightarrow{\alpha} \rho$ *and* $\rho' \models \varphi$ *for some* $\rho' \in \mathcal{C}_{\mathcal{T}}$*.*

For a process $p \in P$ *we take* $p \models \varphi$ *to mean* $(p, \lambda) \models \varphi$*.*

We make use of some standard short-hands for Hennessy-Milner-type logics, such as, $\bot = \neg\top$, $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$, $[\alpha]\varphi = \neg\langle\alpha\rangle(\neg\varphi)$ and $[{\leftarrow}\alpha]\varphi = \neg\langle{\leftarrow}\alpha\rangle(\neg\varphi)$. For a finite set of actions B, we also use the following notations.

$$\langle B\rangle\varphi = \bigvee_{\alpha\in B} \langle\alpha\rangle\varphi \qquad \langle{\leftarrow}B\rangle\varphi = \bigvee_{\alpha\in B} \langle{\leftarrow}\alpha\rangle\varphi$$
$$[B]\varphi = \bigwedge_{\alpha\in B} [\alpha]\varphi \qquad [{\leftarrow}B]\varphi = \bigwedge_{\alpha\in B} [{\leftarrow}\alpha]\varphi$$

It is worth mentioning that the operators $\langle\cdot\rangle$ and $\langle{\leftarrow}\cdot\rangle$ are not entirely symmetric. The future is non-deterministic; the past is, however, always deterministic. This is by design, and we could have chosen to model the past as nondeterministic as well, i.e., to take a possibilistic view where we would consider all possible histories. Overall, the deterministic view is more appropriate for our purposes. See, e.g., [17] for a clear discussion of possible approaches in modelling the past and further references.

3 Decomposing Computations

In this section, following [2, 15, 20], we aim at defining a notion of "formula quotient with respect to a process in a parallel composition" for formulae in $HML_{\leftarrow}$. In our setting, this goal translates into a theorem of the form $\rho \vDash \varphi$ iff $\rho_1 \vDash \varphi/\rho_2$, where ρ, ρ_1, ρ_2 are computations such that ρ is a computation of a "parallel process" that is, in some sense, the "parallel composition" of ρ_1 and ρ_2.

In the standard setting, definitions of "formula quotients" are based on local information that can be gleaned from the operational semantics of the chosen notion of parallel composition operator. In the case of computations, however, such local information does not suffice. A computation arising from the evolution of two processes run in parallel has the form $(p \parallel q, \pi)$, where $p \parallel q$ is a syntactic representation of the initial state and π is the path leading up to the current state. The path π however may involve contributions from both of the parallel components. Separating the contributions of the components for the purposes of decompositional model checking requires us to unzip these paths into separate paths that might have been observed by considering only one argument of the composition. This means that we have to find two paths π_p and π_q such that (p, π_p) and (q, π_q) are, in some sense, independent computations that run in parallel will yield $(p \parallel q, \pi)$.

3.1 CCS Computations and Their Decomposition

For this study, in order to highlight the main ideas and technical tools in our approach, we restrict ourselves to a subset of CCS, namely CCS without renaming, restriction or recursion. (We discuss possible extensions of our results in Section 6.) Processes are thus defined by the following grammar.

$$p, q \quad ::= \quad 0 \mid \alpha.p \mid p + q \mid p \parallel q$$

and their operational semantics is given by the following rules.

$$\frac{}{\alpha.p \xrightarrow{\alpha} p} \qquad \frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{a} q'}$$

$$\frac{p \xrightarrow{\alpha} p'}{p \parallel q \xrightarrow{\alpha} p' \parallel q} \qquad \frac{q \xrightarrow{\alpha} q'}{p \parallel q \xrightarrow{\alpha} p \parallel q'} \qquad \frac{p \xrightarrow{a} p' \quad q \xrightarrow{\bar{a}} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

We write $p \xrightarrow{\alpha} q$ to denote that this transition is provable by these rules. We assume also that $\bar{\cdot} : A \to A$ is a bijective function on action names such that $\bar{\bar{a}} = a$.

The decomposition of a computation resulting from the evolution of two parallel components must retain the information about the order of steps in the interleaved computation. We do so by modelling the decomposition using *stuttering computations*. These are computations that are not only sequences of

transition triplets, but may also involve pseudo-steps labelled with $\dashrightarrow$. Intuitively, $p \dashrightarrow p$ means that process p has remained idle in the last transition performed by a parallel process having p as one of its parallel components. We denote the set of stuttering computations with $\mathcal{C}^*_{\mathcal{T}}$ or simply $\mathcal{C}^*$. For example, the computation

$$(a.0 \parallel b.0, a.0 \parallel b.0 \xrightarrow{a} 0 \parallel b.0 \xrightarrow{b} 0 \parallel 0)$$

is decomposed into the stuttering computations

$$(a.0, a.0 \xrightarrow{a} 0 \dashrightarrow 0) \quad \text{and} \quad (b.0, b.0 \dashrightarrow b.0 \xrightarrow{b} 0).$$

However, the decomposition of a parallel computation is not in general unique, as there may be several possibilities stemming from different synchronisation patterns. For example consider a computation with path $(a.0 + b.0) \parallel (\bar{a}.0 + \bar{b}.0) \xrightarrow{\tau} 0 \parallel 0$. From this computation it is not possible to distinguish if the transition labelled with τ was the result of communication of the a and $\bar{a}$ actions, or of the b and $\bar{b}$ actions. We thus consider *all* possibilities simultaneously, i.e., a decomposition of a computation is actually *a set* of pairs of components.

The following function over paths defines the decomposition of a computation.

$$D(\lambda) = \{(\lambda, \lambda)\}$$

$$D(\pi(p \parallel q \dashrightarrow p \parallel q)) = \{(\mu_1(p \dashrightarrow p), \mu_2(q \dashrightarrow q)) \mid (\mu_1, \mu_2) \in D(\pi)\}$$

$$D(\pi(p \parallel q \xrightarrow{\alpha} p' \parallel q')) = \begin{cases} \{(\mu_1(p \xrightarrow{\alpha} p'), \mu_2(q \dashrightarrow q)) \\ \qquad \mid (\mu_1, \mu_2) \in D(\pi)\} & \text{if } q \equiv q' \\ \{(\mu_1(p \dashrightarrow p), \mu_2(q \xrightarrow{\alpha} q')) \\ \qquad \mid (\mu_1, \mu_2) \in D(\pi')\} & \text{if } p \equiv p' \\ \{(\mu_1(p \xrightarrow{a} p'), \mu_2(q \xrightarrow{\bar{a}} q')) \\ \qquad \mid (\mu_1, \mu_2) \in D(\pi), a \in A, \\ \qquad\quad p \xrightarrow{a} p', q \xrightarrow{\bar{a}} q'\} & \text{otherwise and } \alpha = \tau. \end{cases}$$

Note that if (μ_1, μ_2) is a decomposition of a computation π, then the three computations have the same length. Furthermore $\text{last}(\pi) = \text{last}(\mu_1) \parallel \text{last}(\mu_2)$.

Another notable property of path decomposition is that its inverse is unique, i.e., a pair (μ_1, μ_2) can only be the decomposition of a single path.

Lemma 1. *Let π_1 be a path of a parallel computation and $(\mu_1, \mu_2) \in D(\pi_1)$. If π_2 is a path such that $(\mu_1, \mu_2) \in D(\pi_2)$ also, then $\pi_1 = \pi_2$.*

We now aim at defining the quotient of an $HML_{\leftarrow}$-formula φ with respect to a computation (q, μ_2), written $\varphi/(q, \mu_2)$, in such a way that a property of the form

$$(p \parallel q, \pi) \vDash \varphi \quad \Leftrightarrow \quad (p, \mu_1) \vDash \varphi/(q, \mu_2)$$

holds when $(\mu_1, \mu_2) \in D(\pi)$. However, since we are dealing with sets of decompositions, we need to quantify over these sets. It turns out that a natural way that also gives a strong result is the following. Given that a composed computation satisfies a formula, we prove in Section 4 that one component of *every* decomposition satisfies a formula quotiented with the other component:

$$(p \parallel q, \pi) \vDash \varphi \quad \Rightarrow \quad \forall (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash \varphi / (q, \mu_2).$$

On the other hand, to show the implication from right to left, we need only one witness of a decomposition that satisfies a quotiented formula to deduce that the composed computation satisfies the original one:

$$\exists (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash \varphi / (q, \mu_2) \quad \Rightarrow \quad (p \parallel q, \pi) \vDash \varphi.$$

In order to define the quotienting transformation, we need a logic that allows us to describe properties of computations involving explicit pseudo-steps. To this end, we now extend $HML_{\leftarrow}$ with two additional modal operators.

Definition 4 (Stuttering Hennessy-Milner logic with past). *Consider an LTS $\mathcal{T} = \langle P, A, \rightarrow \rangle$. The set $HML^*_{\leftarrow}(A)$, or simply $HML^*_{\leftarrow}$, of* stuttering Hennessy-Milner logic formulae with past *is defined by the grammar*

$$\varphi, \psi ::= \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \langle \alpha \rangle \varphi \mid \langle \leftarrow \alpha \rangle \varphi \mid \langle \dashrightarrow \rangle \varphi \mid \langle \dashleftarrow \rangle \varphi$$

where $\alpha \in A_\tau$. The satisfaction relation $\vDash^* \subseteq \mathcal{C}^*_{\mathcal{T}} \times HML^*_{\leftarrow}$ *is defined in the same manner as for Hennessy-Milner logic with past, by extending Definition 3 with the following two items.*

- $\rho \vDash^* \langle \dashrightarrow \rangle \varphi$ *iff* $\rho(p \dashrightarrow p) \vDash^* \varphi$ *where* $p = \mathrm{last}(\rho)$.
- $\rho \vDash^* \langle \dashleftarrow \rangle \varphi$ *iff* $\rho' \vDash^* \varphi$ *where* $\rho = \rho'(p \dashrightarrow p)$ *for some* p.

Similarly, $\vDash^* \in P \times HML^*_{\leftarrow}$ is defined by $p \vDash^* \varphi$ if and only if $(p, \lambda) \vDash^* \varphi$.

The satisfaction relations $\vDash^*$ and $\vDash$ coincide over $\mathcal{C}_{\mathcal{T}} \times HML_{\leftarrow}$.

Why are the stutters necessary? One may ask why we need to extend both computations and the logic to include the notion of pseudo-steps. The reason for doing so is to capture information about the interleaving order in component computations. This in turn is necessary because the original logic can differentiate between different interleavings of parallel processes.

For an example, consider the computation $(a.0 \parallel b.0, \pi)$ where

$$\pi = a.0 \parallel b.0 \xrightarrow{a} 0 \parallel b.0 \xrightarrow{b} 0 \parallel 0.$$

Clearly this computation does *not* satisfy the formula $\langle \leftarrow a \rangle \top$.

Another interleaving of the same parallel composition is the computation $(a.0 \parallel b.0, \pi')$ where

$$\pi' = a.0 \parallel b.0 \xrightarrow{b} a.0 \parallel 0 \xrightarrow{a} 0 \parallel 0.$$

This computation, on the other hand, does satisfy $\langle \leftarrow a \rangle \top$. Since the logic can distinguish between different interleaving orders of a parallel computation, it is vital to maintain information about the interleaving order in our decomposition. If the decomposition of the above computations only considered the actions contributed by each component, this information would be lost and the two paths would have the same decomposition. As a result, we could not reasonably expect to test if they satisfy the formula $\langle \leftarrow a \rangle \top$ in a decompositional manner.

4 Decompositional Reasoning

We now define the quotienting construction over formulae structurally. The complete quotienting transformation is given in Table 1, where we assume that $p' = \mathrm{last}(\rho)$. Below we limit ourselves to discussing the quotienting transformation for formulae of the form $\langle \leftarrow \alpha \rangle \varphi$.

To define the transformation for formulae of that form, we look at several cases separately. First we consider the case when ρ has the empty path. In this case it is obvious that no backward step is possible.

$$(\langle \leftarrow \alpha \rangle \varphi)/(p, \lambda) = \bot$$

The second case to consider is when ρ ends with a pseudo-transition. In this case the only possibility is that the other component (the one we are testing) is able to perform the backward transition.

$$(\langle \leftarrow \alpha \rangle \varphi)/\rho'(p' \dashrightarrow p') = \langle \leftarrow \alpha \rangle (\varphi/\rho')$$

The third case applies when ρ does indeed end with the transition we look for. In this case the other component must end with a matching pseudo-transition.

$$(\langle \leftarrow \alpha \rangle \varphi)/\rho'(p'' \xrightarrow{\alpha} p') = \langle \dashleftarrow \rangle (\varphi/\rho') \tag{1}$$

The only remaining case to consider is when ρ ends with a transition different from the one we look for. We split this case further and consider again separately the cases when $\alpha \in A$ and when $\alpha = \tau$. The former case is simple: if ρ indicates that the last transition has a label other than the one specified in the diamond operator, the composite computation cannot satisfy $\langle \leftarrow a \rangle \varphi$ because the other component must have performed a pseudo-step.

$$(\langle \leftarrow a \rangle \varphi)/\rho'(p'' \xrightarrow{\beta} p') = \bot \quad \text{where } a \neq \beta$$

If however the diamond operator mentions a τ transition, then we must look for a transition in the other component that can synchronise with the last one of ρ. Note that this case does not include computations ending with a τ transition, as that case is covered by Equation (1).

$$(\langle \leftarrow \tau \rangle \varphi)/\rho'(p'' \xrightarrow{b} p') = \langle \leftarrow \bar{b} \rangle (\varphi/\rho')$$

$$
\begin{aligned}
\top/\rho &= \top \\
(\varphi_1 \wedge \varphi_2)/\rho &= \varphi_1/\rho \wedge \varphi_2/\rho \\
(\neg\varphi)/\rho &= \neg(\varphi/\rho) \\
(\langle a\rangle\varphi)/\rho &= \langle a\rangle\left(\varphi/\rho(p' \dashrightarrow p')\right) \vee \left(\bigvee_{\rho' : \rho \xrightarrow{a} \rho'} \langle \dashrightarrow \rangle(\varphi/\rho')\right) \\
(\langle \tau\rangle\varphi)/\rho &= \langle \tau\rangle\left(\varphi/\rho(p' \dashrightarrow p')\right) \vee \left(\bigvee_{\rho' : \rho \xrightarrow{\tau} \rho'} \langle \dashrightarrow \rangle(\varphi/\rho')\right) \\
&\qquad \vee \left(\bigvee_{\rho', a : \rho \xrightarrow{a} \rho'} \langle \bar{a}\rangle(\varphi/\rho')\right) \\
(\langle \leftarrow \alpha\rangle\varphi)/(p,\lambda) &= \bot \\
(\langle \leftarrow \alpha\rangle\varphi)/\rho'(p' \dashrightarrow p') &= \langle \leftarrow \alpha\rangle(\varphi/\rho') \\
(\langle \leftarrow \alpha\rangle\varphi)/\rho'(p'' \xrightarrow{\alpha} p') &= \langle \dashleftarrow \rangle(\varphi/\rho') \\
(\langle \leftarrow a\rangle\varphi)/\rho'(p'' \xrightarrow{\beta} p') &= \bot \quad \text{where } a \neq \beta \\
(\langle \leftarrow \tau\rangle\varphi)/\rho'(p'' \xrightarrow{b} p') &= \langle \leftarrow \bar{b}\rangle(\varphi/\rho') \\
(\langle \dashrightarrow \rangle\varphi)/\rho &= \langle \dashrightarrow \rangle\left(\varphi/\rho(p' \dashrightarrow p')\right) \\
(\langle \dashleftarrow \rangle\varphi)/\rho &= \begin{cases} \langle \dashleftarrow \rangle(\varphi/\rho') & \text{if } \rho = \rho'(p' \dashrightarrow p') \\ \bot & \text{otherwise} \end{cases}
\end{aligned}
$$

Table 1. Quotienting transformations of formulae in $HML^*_{\leftarrow}$

This covers all possible cases for $\langle \leftarrow \alpha\rangle\varphi/\rho$.

We are now ready to prove the main theorem in this section, to the effect that the quotienting of a formula φ with respect to a computation ρ is properly defined.

Theorem 1. *For CCS processes p, q and a computation $(p \parallel q, \pi) \in \mathcal{C}(p \parallel q)$ and a formula $\varphi \in HML^*_{\leftarrow}$ we have*

$$(p \parallel q, \pi) \vDash^* \varphi \quad \Rightarrow \quad \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* \varphi/(q, \mu_2) \tag{2}$$

and, conversely,

$$(p \parallel q, \pi) \vDash^* \varphi \quad \Leftarrow \quad \exists(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* \varphi/(q, \mu_2). \tag{3}$$

Theorem 1 uses the existential quantifier in the right-to-left direction. This makes it easy to show that a computation of a process of the form $p \parallel q$ satisfies a formula, given only one witness of a decomposition with one component satisfying

the corresponding quotient formula. Note, however, that the set of decompositions of any given process is never empty, i.e., every parallel computation has a decomposition. This allows us to write the above theorem in a more symmetric form.

Corollary 1. *For CCS processes p, q, a parallel computation $(p \parallel q, \pi)$ and a formula $\varphi \in HML^*_{\leftarrow}$ we have*

$$(p \parallel q, \pi) \vDash^* \varphi \quad \Leftrightarrow \quad \forall (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* \varphi/(q, \mu_2). \tag{4}$$

5 Adding recursion to $HML^*_{\leftarrow}$

In this section, we extend the results from Section 4 to a version of the logic $HML^*_{\leftarrow}$ that includes (formula) variables and a facility for the recursive definition of formulae. Following, e.g., [19], the intended meaning of a formula variable is specified by means of a declaration, i.e., a mapping from variables to formulae, which may themselves contain occurrences of variables. A declaration is nothing but a system of equations over the set of formula variables.

By using the extension of the logic $HML^*_{\leftarrow}$ discussed in this section, we can reason about properties of processes and computations that go beyond one step of lookahead or look-back. For example we can phrase the question "Has the action α ever happened in the past?" as the least model of a suitable recursive logical property.

Definition 5. *Let A be a finite set of actions and let $\mathcal{X}$ be a finite set of identifiers. The set $HML^*_{\leftarrow,\mathcal{X}}(A)$, or simply $HML^*_{\leftarrow,\mathcal{X}}$, is defined by the grammar*

$$\varphi, \psi ::= \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \langle\leftarrow\alpha\rangle\varphi \mid \langle\dashrightarrow\rangle\varphi \mid \langle\dashleftarrow\rangle\varphi \mid X$$

where $X \in \mathcal{X}$.

Definition 6. *Let $\mathcal{X}$ be a finite set of variables. Then a* declaration *over $\mathcal{X}$ is a function $\mathcal{D} : \mathcal{X} \to HML^*_{\leftarrow,\mathcal{X}}$, assigning a formula to each variable contained in $\mathcal{X}$, with the restriction that each occurrence of a variable in a formula in the range of $\mathcal{D}$ is* positive*, i.e., any variable is within the scope of an even number of negations.*

When reasoning about recursive formulae, it is technically convenient to define their meaning (i.e., the set of computations that satisfy them) denotationally. For the sake of clarity, we rephrase Definition 4 in a denotational setting. As it is customary, the following definition makes use of a notion of environment to give meaning to formula variables. An *environment* is a function $\sigma : \mathcal{X} \to \mathcal{P}(\mathcal{C}^*)$. Intuitively, an environment assigns to each variable the set of computations that are assumed to satisfy it. We write $\mathcal{E}_{\mathcal{X}}$ for the set of environments over the set of (formula) variables $\mathcal{X}$. It is well-known that $\mathcal{E}_{\mathcal{X}}$ is a complete lattice when environments are ordered pointwise using set inclusion.

Definition 7 (Denotational semantics of $HML^*_{\leftarrow,\mathcal{X}}$). *Let $\mathcal{T} = \langle P, A, \rightarrow\rangle$ be an LTS. Let φ be a $HML^*_{\leftarrow,\mathcal{X}}$ formula and let σ be an environment. The denotation of φ with respect to σ, written $[\![\varphi]\!]\sigma$, is defined structurally as follows:*

$$\begin{aligned}
[\![\top]\!]\sigma &= \mathcal{C}^*_{\mathcal{T}} \\
[\![\varphi \wedge \psi]\!]\sigma &= [\![\varphi]\!]\sigma \cap [\![\psi]\!]\sigma \\
[\![\neg\varphi]\!]\sigma &= \mathcal{C}^*_{\mathcal{T}} \setminus [\![\varphi]\!]\sigma \\
[\![\langle\alpha\rangle\varphi]\!]\sigma &= \langle\cdot\alpha\cdot\rangle[\![\varphi]\!]\sigma \\
[\![\langle\leftarrow\alpha\rangle\varphi]\!]\sigma &= \langle\cdot\leftarrow\alpha\cdot\rangle[\![\varphi]\!]\sigma \\
[\![\langle\dashrightarrow\rangle\varphi]\!]\sigma &= \langle\cdot\dashrightarrow\cdot\rangle[\![\varphi]\!]\sigma \\
[\![\langle\dashleftarrow\rangle\varphi]\!]\sigma &= \langle\cdot\dashleftarrow\cdot\rangle[\![\varphi]\!]\sigma \quad \textit{and} \\
[\![X]\!]\sigma &= \sigma(X),
\end{aligned}$$

where the operators $\langle\cdot\alpha\cdot\rangle, \langle\cdot\leftarrow\alpha\cdot\rangle, \langle\cdot\dashrightarrow\cdot\rangle, \langle\cdot\dashleftarrow\cdot\rangle : \mathcal{P}(\mathcal{C}^*_{\mathcal{T}}) \rightarrow \mathcal{P}(\mathcal{C}^*_{\mathcal{T}})$ are defined thus:

$$\begin{aligned}
\langle\cdot\alpha\cdot\rangle S &= \{\rho \in \mathcal{C}^*_{\mathcal{T}} \mid \exists\rho' \in S : \rho \xrightarrow{\alpha} \rho'\} \\
\langle\cdot\leftarrow\alpha\cdot\rangle S &= \{\rho \in \mathcal{C}^*_{\mathcal{T}} \mid \exists\rho' \in S : \rho' \xrightarrow{\alpha} \rho\} \\
\langle\cdot\dashrightarrow\cdot\rangle S &= \{\rho \in \mathcal{C}^*_{\mathcal{T}} \mid \exists\rho' \in S : \rho \dashrightarrow \rho'\} \quad \textit{and} \\
\langle\cdot\dashleftarrow\cdot\rangle S &= \{\rho \in \mathcal{C}^*_{\mathcal{T}} \mid \exists\rho' \in S : \rho' \dashrightarrow \rho\}.
\end{aligned}$$

The satisfaction relation $\models_\sigma \subseteq \mathcal{C}^*_{\mathcal{T}} \times HML^*_{\leftarrow,\mathcal{X}}$ is defined by

$$\rho \models_\sigma \varphi \quad \Leftrightarrow \quad \rho \in [\![\varphi]\!]\sigma.$$

It is not hard to see that, for formulae in $HML^*_{\leftarrow}$, the denotational semantics is independent of the chosen environment and is equivalent to the satisfaction relation offered in Definition 4 in the sense made precise by the following lemma.

Lemma 2. *Let $\varphi \in HML^*_{\leftarrow}$ and $\rho \in \mathcal{C}^*_{\mathcal{T}}$. Then $\rho \models^* \varphi$ if and only if $\rho \in [\![\varphi]\!]\sigma$, for any environment σ.*

The semantics of a declaration $\mathcal{D}$ is given by a *model* for it, viz. by an environment σ such that $\sigma(X) = [\![\mathcal{D}(X)]\!]\sigma$, for each variable $X \in \mathcal{X}$. For every declaration there may be a variety of models. However, we are usually interested in either the greatest or least models, since they correspond to safety and liveness properties, respectively. In the light of the positivity restrictions we have placed on the formulae in the range of declarations, each declaration always has least and largest models by Tarski's fixed-point theorem [25]. See, e.g., [1, 19] for details and textbook presentations.

5.1 Decomposition of formulae in $HML^*_{\leftarrow,\mathcal{X}}$

We now turn to the transformation of formulae, so that we can extend Theorem 1 to include formulae from $HML^*_{\leftarrow,\mathcal{X}}$. Our developments in this section are inspired by [13], but the technical details are rather different and more involved.

In Section 4 we defined how a formula φ is quotiented with respect to a computation ρ. In particular, the quotiented formula $\top/\rho$ is $\top$ for any computation ρ. This works well in the non-recursive setting, but there is a hidden assumption that we must expose before tackling recursive formulae. In Theorem 1, the satisfaction relations are actually based on two different transition systems. By way of example, consider the expression on the right-hand side of the theorem, namely

$$\forall(\mu_1, \mu_2) \in D(\pi) : \ (p, \mu_1) \vDash \varphi/(q, \mu_2).$$

When establishing this statement, we have implicitly assumed that we are working within the transition system of computations from p that are *compatible* with the computations from q—i.e., above, μ_1 really is a path that is the counterpart of μ_2 in a decomposition of the path π.

Intuitively, the set of computations that satisfy a quotient formula φ/ρ is the set of computations that *are compatible with* ρ and whose *composition with* ρ satisfies the formula φ. However, defining $\top/\rho = \top$ does not match this intuition, if we take the denotational viewpoint of the formula $\top$ on the right-hand side representing *all* possible computations. In fact, we expect $\top/\rho$ to represent only those computations that are compatible with ρ. We formalize the notion of pairs of compatible computations and refine our definition of $\top/\rho$.

Definition 8. *Paths μ_1 and μ_2 are* compatible *with each other if and only if they have the same length and one of the following holds if they are non-empty.*

- *If $\mu_1 = \mu_1'(p'' \xrightarrow{\tau} p')$ then $\mu_2 = \mu_2'(q' \dashrightarrow q')$ and μ_1' and μ_2' are compatible.*
- *If $\mu_1 = \mu_1'(p'' \xrightarrow{a} p')$ then either $\mu_2 = \mu_2'(q'' \xrightarrow{\bar{a}} q')$ or $\mu_2 = \mu_2'(q' \dashrightarrow q')$; and in both cases μ_1' and μ_2' are compatible.*
- *If $\mu_1 = \mu_1'(p'' \dashrightarrow p')$ then either $\mu_2 = \mu_2'(q'' \xrightarrow{\alpha} q')$, for some action α, or $\mu_2 = \mu_2'(q' \dashrightarrow q')$; and in both cases μ_1' and μ_2' are compatible.*

We say that two computations are compatible with each other if their paths are compatible.

We now revise our transformation of the formula $\top$. We want $\top/\rho$ to be a formula that is satisfied by the set of all computations that are compatible with ρ. It turns out this can be expressed in $HML^*_{\leftarrow}$ as described below.

Definition 9. *Let π be a path of transitions in the LTS $\mathcal{T} = \langle P, A, \rightarrow \rangle$. Then the $HML^*_{\leftarrow}$ formula $\top_\pi$ is defined as follows.*

$$\begin{aligned}
\top_\lambda &= [\leftarrow A_\tau]\bot \wedge [\dashleftarrow]\bot \\
\top_{\pi'(p \xrightarrow{\tau} p')} &= \langle \dashleftarrow \rangle \top_{\pi'} \\
\top_{\pi'(p \xrightarrow{a} p')} &= \langle \leftarrow \bar{a} \rangle \top_{\pi'} \vee \langle \dashleftarrow \rangle \top_{\pi'} \\
\top_{\pi'(p \dashrightarrow p')} &= \langle \leftarrow A_\tau \rangle \top_{\pi'} \vee \langle \dashleftarrow \rangle \top_{\pi'}
\end{aligned}$$

Our reader may notice that this is a rewording of Definition 8, and it is easy to see that the computations satisfying $\top_\pi$ are exactly the computations which have paths compatible with π. Now the revised transformation of $\top$ is

$$\top/(p, \pi) = \top_\pi, \tag{5}$$

which matches our intuition. For the constructs in the logic $HML^*_{\leftarrow}$, we can reuse the transformation defined in Section 4. We therefore limit ourselves to highlighting how to quotient formulae of the form X. However, instead of decomposing formulae of this form, we treat the quotient X/ρ as a variable, i.e., we use the set $\mathcal{X} \times \mathcal{C}$ as our set of variables. The intuitive idea of such variables is the following,

$$(p, \mu_1) \vDash_{\sigma'} X/(q, \mu_2) \quad \Leftrightarrow \quad (p \parallel q, \pi) \vDash_{\sigma} X \quad \Leftrightarrow \quad (p \parallel q, \pi) \in \sigma(X)$$

where σ is an environment for a declaration $\mathcal{D}$ over the variables $\mathcal{X}$, σ' is an environment for a declaration $\mathcal{D}'$ over the variables $\mathcal{X} \times \mathcal{C}$, and $(\mu_1, \mu_2) \in D(\pi)$. We explain below the relation between $\mathcal{D}$ and $\mathcal{D}'$ as well as the one between σ and σ'.

Formally, the variables used in quotienting our logic are pairs $(X, \rho) \in \mathcal{X} \times \mathcal{C}$. Formulae of the form X are simply rewritten as follows:

$$X/\rho = (X, \rho),$$

where the X/ρ on the left-hand side denotes the transformation (as in Section 4) and the pair on the right-hand side is the variable in our adapted logic. When there is no risk of ambiguity, we simply use the notation X/ρ to represent *the variable* (X, ρ).

Transformation of declarations Generating the transformed declaration $\mathcal{D}'$ from a declaration $\mathcal{D}$ is done as follows:

$$\mathcal{D}'(X/\rho) = \mathcal{D}(X)/\rho. \tag{6}$$

Note that the rewritten formula on the right-hand side may introduce more variables which obtain their values in $\mathcal{D}'$ in the same manner.

Transformation of environments The function Φ maps environments over $\mathcal{X}$ to environments over $\mathcal{X} \times \mathcal{C}$ thus:

$$\begin{aligned} \sigma'(X/(q, \mu_2)) &= \Phi(\sigma)(X/(q, \mu_2)) \\ &= \{(p, \mu_1) \mid (p \parallel q, \pi) \in \sigma(X) \text{ for some } \pi \text{ with } (\mu_1, \mu_2) \in D(\pi)\}. \end{aligned}$$

Our order of business now is to show that if σ is the least (respectively, largest) model for a declaration $\mathcal{D}$, then σ' is the least (respectively, largest) model for $\mathcal{D}'$ and vice versa. In particular, we show that there is a bijection relating models of $\mathcal{D}$ and models of $\mathcal{D}'$, based on the mapping Φ. First we define its inverse. Consider the function Ψ, which maps an environment over $\mathcal{X} \times \mathcal{C}$ to one over $\mathcal{X}$.

$$\Psi(\sigma')(X) = \{(p \parallel q, \pi) \mid \forall (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \in \sigma'(X/(q, \mu_2))\}$$

It is not hard to see that Φ and Ψ are both monotonic.

We now use the model transformation functions Φ and Ψ to prove an extended version of Theorem 1.

Theorem 2. *Let p, q be CCS processes, $(p \parallel q, \pi) \in \mathcal{C}^*(p \parallel q)$. For a formula $\varphi \in HML^*_{\leftarrow,\mathcal{X}}$ and an environment σ, we have*

$$(p \parallel q, \pi) \vDash_\sigma \varphi \quad \Leftrightarrow \quad \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash_{\Phi(\sigma)} \varphi/(q, \mu_2). \tag{7}$$

Conversely, for an environment σ',

$$(p \parallel q, \pi) \vDash_{\Psi(\sigma')} \varphi \quad \Leftrightarrow \quad \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash_{\sigma'} \varphi/(q, \mu_2). \tag{8}$$

We can now show that the functions Φ and Ψ are inverses of each other.

Lemma 3. *$\Psi \circ \Phi = \mathrm{id}_{\mathcal{E}_\mathcal{X}}$ and $\Phi \circ \Psi = \mathrm{id}_{\mathcal{E}_{\mathcal{X}\times\mathcal{C}}}$.*

This means that Φ is a bijection between the collections of environments over the variable spaces $\mathcal{X}$ and $\mathcal{X} \times \mathcal{C}$, and Ψ is its inverse. The last theorem of this section establishes soundness of the decompositional reasoning for $HML^*_{\leftarrow,\mathcal{X}}$ by showing that Φ and Ψ preserve models of $\mathcal{D}$ and $\mathcal{D}'$, respectively.

Theorem 3. *Let $\mathcal{D}$ be a declaration over $\mathcal{X}$, and let $\mathcal{D}'$ be its companion declaration over $\mathcal{X} \times \mathcal{C}$ defined by* (6)*. If σ is a model for $\mathcal{D}$, then $\Phi(\sigma)$ is a model for $\mathcal{D}'$. Moreover, if σ' is a model for $\mathcal{D}'$, then $\Psi(\sigma')$ is a model for $\mathcal{D}$.*

Theorem 3 allows us to use decompositional reasoning for $HML^*_{\leftarrow,\mathcal{X}}$. Assume, for example, that we want to find the least model for a declaration $\mathcal{D}$. We start by constructing the declaration $\mathcal{D}'$ defined by (6). Next, we find the least model $\sigma'_{\min}$ of $\mathcal{D}'$ using standard fixed-point computations. (See, e.g., [1] for a textbook presentation.) We claim that $\Psi(\sigma'_{\min})$ is the least model of the declaration $\mathcal{D}$. Indeed, let σ be any model of $\mathcal{D}$. Then, by the above theorem, $\Phi(\sigma)$ is a model of $\mathcal{D}'$ and thus $\sigma'_{\min} \subseteq \Phi(\sigma)$ holds, where $\subseteq$ is lifted pointwise to environments. Then the monotonicity of Ψ and Lemma 3 ensure that

$$\Psi(\sigma'_{\min}) \subseteq \Psi(\Phi(\sigma)) = \sigma.$$

To conclude, note that $\Psi(\sigma'_{\min})$ is a model of $\mathcal{D}$ by the above theorem.

6 Extensions and further related work

In this paper, we have developed techniques that allow us to apply decompositional reasoning for history-based computations over CCS and Hennessy-Milner logic with past modalities. Moreover, we extended the decomposition theorem to a recursive extension of that logic. In the decomposition of computations that is at the heart of our approach, we rely on some specific properties of CCS at the syntactic level, namely to detect which rule of the parallel operator was applied. By tagging a transition with its proof [5, 7], or even just with the last rule used in the proof, we could eliminate this restriction and extend our approach to other languages involving parallel composition. Another possibility is to construct a rule format that guarantees the properties we use at a more general level, inspired by the work of [8].

In this work we have only considered contexts built using parallel composition. However, decompositional results have been shown for the more general setting of *process contexts* [20] and for rule formats [4, 8]. In that work, one considers, for example, a unary context $C[\cdot]$ (a process term with a *hole*) and a process p with which to instantiate the context. A property of the instantiated context $C[p]$ can then be transformed into an equivalent property of p, where the transformation depends on C. As the state space explosion of model-checking problems is often due to the use of the parallel construct, we consider our approach a useful first step towards a full decomposition result for more general contexts. In general, the decomposition of computations will be more complex for general contexts.

The initial motivation for this work was the application of epistemic logic to behavioural models, following the lines of [6]. We would therefore like to extend our results to logics that include epistemic operators, reasoning about the knowledge of agents observing a running system. This work depends somewhat on the results presented in Section 5.

As we already mentioned in the introduction, there is by now a substantial body of work on temporal and modal logics with past operators. A small sample is given by the papers [11, 16, 26]. Of particular relevance for our work in this paper is the result in [16] to the effect that Hennessy-Milner logic with past modalities can be translated into ordinary Hennessy-Milner logic. That result, however, is only proved for the version of the logic without recursion and does not directly yield a quotienting construction for the logics we consider in this paper.

References

1. L. Aceto, A. Ingolfsdottir, K. G. Larsen, and J. Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, August 2007.
2. H. R. Andersen. Partial model checking (extended abstract). In *LICS*, pages 398–407. IEEE Computer Society, 1995.
3. H. R. Andersen, C. Stirling, and G. Winskel. A compositional proof system for the modal mu-calculus. In *LICS*, pages 144–153. IEEE Computer Society, 1994.
4. B. Bloom, W. Fokkink, and R. J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Trans. Comput. Log.*, 5(1):26–78, 2004.
5. G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, 11(4):433–452, 1988.
6. F. Dechesne, M. Mousavi, and S. Orzan. Operational and epistemic approaches to protocol analysis: Bridging the gap. In *Proceedings of the 14th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'07)*, volume 4790 of *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, Berlin, Germany, 2007.
7. P. Degano and C. Priami. Proved trees. In W. Kuich, editor, *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 629–640. Springer, 1992.
8. W. Fokkink, R. J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic by structural operational semantics. *Theoretical Computer Science*, 354(3):421–440, 2006.

9. D. Giannakopoulou, C. S. Pasareanu, and H. Barringer. Component verification with automatically generated assumptions. *Automated Software Engineering*, 12(3):297–320, 2005.
10. J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.
11. M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, 67(1-3):23–52, 1985.
12. T. A. Henzinger, O. Kupferman, and S. Qadeer. From pre-historic to post-modern symbolic model checking. *Formal Methods in System Design*, 23(3):303–327, 2003.
13. A. Ingólfsdóttir, J. C. Godskesen, and M. Zeeberg. Fra Hennessy-Milner logik til CCS-processer. Technical report, Aalborg Universitetscenter, 1987.
14. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
15. F. Laroussinie and K. G. Larsen. Compositional model checking of real time systems. In I. Lee and S. A. Smolka, editors, *CONCUR*, volume 962 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 1995.
16. F. Laroussinie, S. Pinchinat, and P. Schnoebelen. Translations between modal logics of reactive systems. *Theor. Comput. Sci.*, 140(1):53–71, 1995.
17. F. Laroussinie and P. Schnoebelen. Sepcification in CTL+past for verification in CTL. *Information and Computation*, 156(1):236–263, 2000.
18. K. G. Larsen. *Context-dependent bisimulation between processes*. PhD thesis, Department of Computer Science, University of Edinburgh, 1986.
19. K. G. Larsen. Proof systems for satisfiability in Hennessy–Milner logic with recursion. *Theoretical Comput. Sci.*, 72(2–3):265–288, 23 May 1990.
20. K. G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, 1991.
21. O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In R. Parikh, editor, *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.
22. I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. In L. Aceto and A. Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2006.
23. A. K. Simpson. Sequent calculi for process verification: Hennessy-Milner logic for an arbitrary GSOS. *Journal of Logic and Algebraic Programming*, 60-61:287–322, 2004.
24. C. Stirling. A complete compositional model proof system for a subset of CCS. In W. Brauer, editor, *ICALP*, volume 194 of *Lecture Notes in Computer Science*, pages 475–486. Springer, 1985.
25. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
26. M. Y. Vardi. Reasoning about the past with two-way automata. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.
27. G. Winskel. A complete system for SCCS with modal assertions. In S. N. Maheshwari, editor, *FSTTCS*, volume 206 of *Lecture Notes in Computer Science*, pages 392–410. Springer, 1985.
28. G. Xie and Z. Dang. Testing systems of concurrent black-boxes—an automata-theoretic and decompositional approach. In W. Grieskamp and C. Weise, editors, *Post-Proceedings of the 5th International Workshop on Formal Approaches to Software Testing (FATES'05)*, volume 3997 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2006.

A Proofs for Section 4

Proof of Lemma 1. We start by noting that π_1 and π_2 cannot differ in length, as they are both equal in length to μ_1 (and μ_2). We apply induction on their common length.

If both are empty, $\pi_1 = \pi_2 = \lambda$, then there is nothing to prove. Now assume they are non-empty and that

$$\pi_1 = \pi_1'(p_1' \parallel q_1' \ R_1 \ p_1 \parallel q_1)$$
$$\pi_2 = \pi_2'(p_2' \parallel q_2' \ R_2 \ p_2 \parallel q_2)$$

where R_1, R_2 are relations of the form $\xrightarrow{\alpha}$ or $\dashrightarrow$. The induction hypothesis states that $\pi_1' = \pi_2'$, which also means that $p_1' = p_2'$ and $q_1' = q_2'$. Furthermore, $p_1 = p_2$ and $q_1 = q_2$. Thus we only need to show that the final steps coincide also, i.e. that $R_1 = R_2$. The proof proceeds by case analysis on the last steps of μ_1 and μ_2.

- If both μ_1 and μ_2 end with a pseudo-step, then we see from the definition of D that both R_1 and R_2 must be pseudo-transitions.
- If only one of μ_1 and μ_2 ends with a pseudo-step, then the action of the other one must be the same as the last action of both π and π'.
- If both μ_1 and μ_2 end with a proper transition, we note that by the definition of D the actions must complement each other. Then the last step of both π and π' must thus be labelled with τ.
- If both μ_1 and μ_2 end with a proper transition, we note that by the definition of D the actions must complement each other. Then the last step of both π and π' must thus be labelled with τ.

This covers all the cases and thus we have shown that $R_1 = R_2$, $p_1 = p_2$ and $q_1 = q_2$. Coupled with the induction hypothesis, this means that $\pi = \pi'$. □

Lemma 4. *If $p \parallel q \xrightarrow{\alpha} p' \parallel q'$ where $p \not\equiv p'$ and $q \not\equiv q'$ then $\alpha = \tau$.*

Proof of Lemma 4. Consider the proof tree for the transition $p \parallel q \xrightarrow{\alpha} p' \parallel q'$ and, in particular, the last rule used in the proof. This rule can be one of the three rules for the parallel operator. The first two, where only one component advances, are ruled out since then either $p \equiv p'$ or $q \equiv q'$ must hold. Therefore the last rule used in the proof must be the communication rule, in which case the label of the proved transition can only be τ. □

Lemma 5. *Let p, q be processes, $(p \parallel q, \pi) \in \mathcal{C}(p \parallel q)$ and $(\mu_1, \mu_2) \in D(\pi)$.*
(i) If $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$ then there exists a pair $(\mu_1', \mu_2') \in D(\pi')$ such that one of the following holds.

1. *$(p, \mu_1) \xrightarrow{\alpha} (p, \mu_1')$ and $(q, \mu_2) \dashrightarrow (q, \mu_2')$,*
2. *$(p, \mu_1) \dashrightarrow (p, \mu_1')$ and $(q, \mu_2) \xrightarrow{\alpha} (q, \mu_2')$ or*
3. *$\alpha = \tau$, $(p, \mu_1) \xrightarrow{a} (p, \mu_1')$ and $(q, \mu_2) \xrightarrow{\bar{a}} (q, \mu_2')$ for some $a \in A$.*

(ii) Symmetrically,

1. *If there exists a* μ_1' *s.t.* $(p, \mu_1) \xrightarrow{\alpha} (p, \mu_1')$ *then there exists a* π' *s.t.* $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$ *and* $(\mu_1', \mu_2(q' \dashrightarrow q')) \in D(\pi')$ *where* $q' = \text{last}(\mu_2)$.
2. *If there exists a* μ_2' *s.t.* $(q, \mu_2) \xrightarrow{\alpha} (q, \mu_2')$ *then there exists a* π' *s.t.* $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$ *and* $(\mu_1(p' \dashrightarrow p'), \mu_2') \in D(\pi')$ *where* $p' = \text{last}(\mu_1)$.
3. *If there exist* μ_1' *and* μ_2' *s.t.* $(p, \mu_1) \xrightarrow{a} (p, \mu_1')$ *and* $(q, \mu_2) \xrightarrow{\bar{a}} (q, \mu_2')$ *for some* $a \in A$, *then there exists* π' *s.t.* $(p \parallel q, \pi) \xrightarrow{\tau} (p \parallel q, \pi')$ *and* $(\mu_1', \mu_2') \in D(\pi')$.

Proof of Lemma 5. (i) Assume that $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$ and let $(\mu_1, \mu_2) \in D(\pi)$. This means there exist processes p', q', p'', q'' with $\pi' = \pi(p'' \parallel q'' \xrightarrow{\alpha} p' \parallel q')$, $p'' = \text{last}(\mu_1), q'' = \text{last}(\mu_2)$. Since $p'' \parallel q'' \not\equiv p' \parallel q'$ we observe that $p'' \equiv p'$ and $q'' \equiv q'$ cannot hold simultaneously, so we consider the remaining cases.

1. $p'' \not\equiv p'$ and $q'' \equiv q'$. In this case the transition $p'' \parallel q' \xrightarrow{\alpha} p' \parallel q'$ was proven using the first rule for $\parallel$. Its only premise must hold, namely $p'' \xrightarrow{\alpha} p'$. We therefore let $\mu_1' = \mu_1(p'' \xrightarrow{\alpha} p')$ and $\mu_2' = \mu_2(q' \dashrightarrow q')$. From the inductive definition of D it is easy to see that $(\mu_1', \mu_2') \in D(\pi')$.
2. $p'' \equiv p'$ and $q'' \not\equiv q'$. This case is entirely symmetric to the previous one where the proof is based on the second rule for $\parallel$.
3. $p'' \not\equiv p'$ and $q'' \not\equiv q'$. Here the proof of the transition $p'' \parallel q'' \xrightarrow{\alpha} p' \parallel q'$ must be based on the third rule for $\parallel$, namely the communication rule and $\alpha = \tau$, as seen by Lemma 4. By the premises of this rule there exists an $a \in A$ such that $p'' \xrightarrow{a} p'$ and $q'' \xrightarrow{\bar{a}} q'$. We simply let $\mu_1' = \mu_1(p'' \xrightarrow{a} p')$ and $\mu_2' = \mu_2(q'' \xrightarrow{\bar{a}} q')$. Again it is clear from the definition of D that $(\mu_1', \mu_2') \in D(\pi')$.

(ii) The construction of π' in all cases is straightforward and unique (cfr. Lemma 1). The rest is simple to check with the definition of D. □

Lemma 6. *Let* $(p \parallel q, \pi) \in \mathcal{C}(p \parallel q)$ *with* π *non-empty and* $(\mu_1, \mu_2) \in D(\pi)$. *Let* π', μ_1' *and* μ_2' *be the prefixes of length* $|\pi| - 1$ *of* π, μ_1 *and* μ_2 *respectively. Then* $(\mu_1', \mu_2') \in D(\pi')$.

This lemma follows directly from the definition of D.

Proof of Theorem 1. We prove both implications simultaneously by induction on the structure of φ. In the following text, the terms *"the left-hand side"* and *"the right-hand side"* refer respectively to the left- and right-hand sides of the above implications where the quantifier used in the right-hand side will be made clear by the context.

Case $\varphi = \top$ Then $\varphi/(q, \mu_2) = \top$ and both sides of both (2) and (3) are trivially satisfied.

Case $\varphi = \psi_1 \wedge \psi_2$

($\Rightarrow$) First assume $(p \parallel q, \pi) \vDash^* \psi_1 \wedge \psi_2$ and let $(\mu_1, \mu_2) \in D(\pi)$. Since both ψ_1 and ψ_2 are smaller than φ and both are satisfied by $(p \parallel q, \pi)$ we have

by induction that $(p, \mu_1) \models^* \psi_i/(q, \mu_2)$ for $i \in \{1, 2\}$. Since $\varphi/(q, \mu_2) = (\psi_1 \wedge \psi_2)/(q, \mu_2) = \psi_1/(q, \mu_2) \wedge \psi_2/(q, \mu_2)$ we obtain $(p, \mu_1) \models^* \varphi/(q, \mu_2)$.

($\Leftarrow$) Now assume the right side of (3),

$$\exists(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \models^* (\psi_1 \wedge \psi_2)/(q, \mu_2).$$

By definition the formula is equal to $\psi_1/(q, \mu_1) \wedge \psi_2/(q, \mu_2)$. By induction $(p \parallel q, \pi)$ satisfies both ψ_1 and ψ_2 and thus also $\psi_1 \wedge \psi_2 = \varphi$.

Case $\varphi = \neg\psi$

($\Rightarrow$) First assume the left side $(p \parallel q, \pi) \models^* \neg\psi$. Assume towards contradiction that there does exist a decomposition $(\mu_1', \mu_2') \in D(\pi)$ such that $(p, \mu_1') \models^* \psi/(q, \mu_2')$. Then by induction (3) gives $(p \parallel q, \pi) \models^* \psi$, which is in direct contradiction with our assumption. Since no such decomposition can exist, it holds for all $(\mu_1, \mu_2) \in D(\pi)$ that $(p, \mu_1) \models^* \neg\psi/(q, \mu_2) = \varphi/(q, \mu_2)$.

($\Leftarrow$) Assume the right side of (3), namely there exists a decomposition $(\mu_1, \mu_2) \in D(\pi)$ such that $(p, \mu_1) \models^* \neg\psi/(q, \mu_2)$. Assume, again towards a contradiction, that $(p \parallel q, \pi) \models^* \psi$. By induction, (2) then gives that for all $(\mu_1', \mu_2') \in D(\pi)$, $(p, \mu_1') \models^* \psi/(q, \mu_2')$. In particular, this holds for the decomposition (μ_1, μ_2), which contradicts our assumption. Therefore we must have that $(p \parallel q, \pi) \models^* \neg\psi = \varphi$.

Case $\varphi = \langle\alpha\rangle\psi$

($\Rightarrow$) Again, first assume the left side and take $(\mu_1, \mu_2) \in D(\pi)$. Then there exists a computation $(p \parallel q, \pi')$ s.t. $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$ and $(p \parallel q, \pi') \models^* \psi$. By part (i) of Lemma 5 there exists a pair $(\mu_1', \mu_2') \in D(\pi')$. Since ψ is a subformula of φ we have by induction that

$$(p, \mu_1') \models^* \psi/(q, \mu_2') \tag{9}$$

Lemma 5 also states that one of the following three cases holds.

1. $(p, \mu_1) \xrightarrow{\alpha} (p, \mu_1')$ and $(q, \mu_2) \dashrightarrow (q, \mu_2')$. From (9) we have that $(p, \mu_1) \models^* \langle\alpha\rangle(\psi/(q, \mu_2'))$ and since the formula $\langle\alpha\rangle(\psi/(q, \mu_2'))$ is the first clause of the disjunction defining $\varphi/(q, \mu_2)$ then also $(p, \mu_1) \models^* \varphi/(q, \mu_2)$.
2. $(p, \mu_1) \dashrightarrow (p, \mu_1')$ and $(q, \mu_2) \xrightarrow{\alpha} (q, \mu_2')$. Again from (9) we have that $(p, \mu_1) \models^* \langle\dashrightarrow\rangle(\psi/(q, \mu_2'))$, and again the formula $\langle\dashrightarrow\rangle(\psi/(q, \mu_2'))$ is a clause of the disjunction defining $\varphi/(q, \mu_2)$ so $(p, \mu_1) \models^* \varphi/(q, \mu_2)$.
3. $\alpha = \tau$, $(p, \mu_1) \xrightarrow{a} (p, \mu_1')$ and $(q, \mu_2) \xrightarrow{\bar{a}} (q, \mu_2')$ for some $a \in A$. Then the disjunction $\varphi/(q, \mu_2)$ has a clause $\langle a\rangle\,(\psi/(q, \mu_2'))$ (note that $\bar{\bar{a}} = a$). By (9) we get that $(p, \mu_1) \models^* \varphi/(q, \mu_2)$.

In all cases the result is the same, namely $(p, \mu_1) \models^* \varphi/(q, \mu_2)$ which is what we wanted to prove.

($\Leftarrow$) Now assume the right side of (3), i.e. there exists a $(\mu_1, \mu_2) \in D(\pi)$ such that that $(p, \mu_1) \models^* \langle\alpha\rangle\psi/(q, \mu_2)$. We know $\langle\alpha\rangle\psi/(q, \mu_2)$ is a disjunction of one or more clauses so (p, μ_1) must satisfy at least one of them. Each clause has one

of three forms, and we analyze the possible cases. Let φ' be a clause that (p, μ_1) satisfies.

1. Assume that $\varphi' = \langle\alpha\rangle\,(\psi/(q, \mu_2)(q' \dashrightarrow q'))$ where $q' = \text{last}(q, \mu_2)$. Then there is a μ_1' such that $(p, \mu_1) \xrightarrow{\alpha} (p, \mu_1')$ and $(p, \mu_1') \vDash^* \psi/(q, \mu_2(q' \dashrightarrow q'))$. If we let $\mu_2' = \mu_2(q' \dashrightarrow q')$ then part (ii) of Lemma 5 gives that there exists a π' with $(\mu_1', \mu_2') \in D(\pi')$ and $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$. Since $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$ then by induction, since ψ is smaller than φ, $(p \parallel q, \pi') \vDash^* \psi$. This in turn means that $(p \parallel q, \pi) \vDash^* \langle\alpha\rangle\psi = \varphi$.
2. Assume that $\varphi' = \langle\dashrightarrow\rangle\,(\psi/(q, \mu_2'))$ for some μ_2' such that $(q, \mu_2) \xrightarrow{\alpha} (q, \mu_2')$. Let $\mu_1' = \mu_1(p' \dashrightarrow p')$ where $p' = \text{last}(p, \mu_1)$. Lemma 5 gives the existence of π' with $(\mu_1', \mu_2') \in D(\pi')$ and $(p \parallel q, \pi) \xrightarrow{\alpha} (p \parallel q, \pi')$. Since $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$ then by induction $(p \parallel q, \pi') \vDash^* \psi$ and thus $(p \parallel q, \pi) \vDash^* \langle\alpha\rangle\psi = \varphi$.
3. Assume that $\alpha = \tau$ and $\varphi' = \langle\bar{a}\rangle\,(\psi/(q, \mu_2'))$ for some μ_2' s.t. $(q, \mu_2) \xrightarrow{a} (q, \mu_2')$ and $a \in A$. This means there is a μ_1' with $(p, \mu_1) \xrightarrow{\bar{a}} (p, \mu_1')$. Lemma 5 then says that there exists π' with $(p \parallel q, \pi) \xrightarrow{\tau} (p \parallel q, \pi')$ and $(\mu_1', \mu_2') \in D(\pi')$. Since $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$ then by induction $(p \parallel q, \pi') \vDash^* \psi$ and thus $(p \parallel q, \pi) \vDash^* \langle\tau\rangle\psi = \varphi$.

In all cases we obtain what we wanted to prove, namely $(p \parallel q, \pi) \vDash^* \varphi$.

Case $\varphi = \langle\leftarrow\alpha\rangle\psi$

($\Rightarrow$) Assume that $(p \parallel q, \pi) \vDash^* \langle\leftarrow\alpha\rangle\psi$ and take $(\mu_1, \mu_2) \in D(\pi)$. Since $(p \parallel q, \pi') \xrightarrow{\alpha} (p \parallel q, \pi)$ for some π' such that $(p \parallel q, \pi') \vDash^* \psi$, there exist processes p', q', p'', q'' such that $\pi = \pi'(p'' \parallel q'' \xrightarrow{\alpha} p' \parallel q')$. By analysing the definition of D, we can gain some information about μ_1 and μ_2, in particular by comparing p'' to p' and q'' to q'. Since $p'' \parallel q'' \not\equiv p' \parallel q'$ we must consider three cases.

1. $p'' \not\equiv p'$ and $q'' \equiv q'$. Then $(\mu_1, \mu_2) = (\mu_1'(p'' \xrightarrow{\alpha} p'), \mu_2'(q' \dashrightarrow q'))$ for some $(\mu_1', \mu_2') \in D(\pi')$. Given this form of μ_2 we also know that $(\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2) = \langle\leftarrow\alpha\rangle\,(\psi/(q, \mu_2'))$. Since $(p \parallel q, \pi') \vDash^* \psi$, we get by induction that $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$, which in turn means that $(p, \mu_1) \vDash^* \langle\leftarrow\alpha\rangle\,(\psi/(q, \mu_2))$ and since the last step of μ_2 is a pseudo-step, $\langle\leftarrow\alpha\rangle\,(\psi/(q, \mu_2)) = (\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2)$.
2. $p'' \equiv p'$ and $q'' \not\equiv q'$. In this case $(\mu_1, \mu_2) = (\mu_1'(p' \dashrightarrow p'), \mu_2'(q'' \xrightarrow{\alpha} q'))$ where $(\mu_1', \mu_2') \in D(\pi')$. This form of μ_2 means that $\langle\leftarrow\alpha\rangle\psi/(q, \mu_2) = \langle\dashleftarrow\rangle\,(\psi/(q, \mu_2'))$. By induction, the fact that $(p \parallel q, \pi') \vDash^* \psi$ gives that $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$, so since $\mu_2 = \mu_2'(q'' \xrightarrow{\alpha} q')$ holds, then we have that $(p, \mu_1) \vDash^* \langle\dashleftarrow\rangle\,(\psi/(q, \mu_2')) = (\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2)$.
3. $p'' \not\equiv p'$ and $q'' \not\equiv q'$. By Lemma 4 α must be equal to τ. Thus we have that $(\mu_1, \mu_2) = (\mu_1'(p'' \xrightarrow{a} p'), \mu_2'(q'' \xrightarrow{\bar{a}} q'))$ for some $a \in A$ and $(\mu_1', \mu_2') \in D(\pi')$. This also means that $(\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2) = \langle\leftarrow a\rangle\,(\psi/(q, \mu_2'))$. Since $(p \parallel q, \pi') \vDash^* \psi$ we again have by induction that $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$. We therefore obtain that $(p, \mu_1) \vDash^* \langle\leftarrow a\rangle\,(\psi/(q, \mu_2')) = (\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2)$.

In all cases we obtain the same result, namely $(p, \mu_1) \vDash^* (\langle\leftarrow\alpha\rangle\psi)/(q, \mu_2) = \varphi/(q, \mu_2)$.

($\Leftarrow$) Now assume that there is $(\mu_1, \mu_2) \in D(\pi)$ s.t. $(p, \mu_1) \vDash^* (\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2)$. This means that μ_1 and μ_2 are non-empty. By comparing α with the last transition of μ_2 we can infer the form of $(\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2)$.

- If the last transition of μ_2 is a $\dashrightarrow$ transition, i.e. $\mu_2 = \mu_2'(q' \dashrightarrow q')$ for some μ_2 and $q' = \text{last}(q, \mu_2)$, then we know that $(\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2) = \langle \leftarrow \alpha \rangle (\psi/(q, \mu_2'))$. By our assumption this is satisfied by (p, μ_1) so there exists a μ_1' s.t. $(p, \mu_1') \xrightarrow{\alpha} (p, \mu_1)$ and $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$. Let π' be π without the last transition (note that π is non-empty since μ_1 and μ_2 are). By Lemma 6 $(\mu_1', \mu_2') \in D(\pi')$ and by induction we have that $(p \parallel q, \pi') \vDash^* \psi$. From the definition of D we can also see that the last transition of π can only be $\xrightarrow{\alpha}$. Thus $(p \parallel q, \pi') \xrightarrow{\alpha} (p \parallel q, \pi)$ so $(p \parallel q, \pi) \vDash^* \langle \leftarrow \alpha \rangle \psi$.
- If the last transition of μ_2 is an $\xrightarrow{\alpha}$ transition, i.e. one having the same label as the formula is testing for, then $(\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2) = \langle \dashleftarrow \rangle (\psi/(q, \mu_2'))$ where μ_2' is μ_2 without the last transition. Note that $(q, \mu_2') \xrightarrow{\alpha} (q, \mu_2)$. Since (p, μ_1) satisfies this formula there is a μ_1' s.t. $(p, \mu_1') \dashrightarrow (p, \mu_1)$ and $(p, \mu_1) \vDash^* \psi/(q, \mu_2')$. By Lemma 6 $(\mu_1', \mu_2') \in D(\pi')$ where π' is again π without the last transition. Also again, we can see from the definition of D that $(p \parallel q, \pi') \xrightarrow{\alpha} (p \parallel q, \pi)$. By induction it thus holds that $(p \parallel q, \pi') \vDash^* \psi$ and so $(p \parallel q, \pi) \vDash^* \langle \leftarrow \alpha \rangle \psi$.
- The only remaining case to consider is when μ_2 ends with a transition $\xrightarrow{\beta}$ where $\beta \neq \alpha$. Then α can only be τ, since otherwise the formula $(\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2)$ equals $\bot$, which contradicts our assumption that (p, μ_1) satisfies it. Since $\beta \neq \alpha = \tau$ we also know β must be some label $a \in A$. This means that $(\langle \leftarrow \alpha \rangle \psi)/(q, \mu_2) = \langle \leftarrow \bar{a} \rangle (\psi/(q, \mu_2'))$ where μ_2' is yet again μ_2 without the last transition. Since (p, μ_1) satisfies this formula, there is a μ_1' s.t. $(p, \mu_1') \xrightarrow{\bar{a}} (p, \mu_1)$ and $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$. By Lemma 6, $(\mu_1', \mu_2') \in D(\pi')$ where π' is π without the last transition. By induction, $(p \parallel q, \pi') \vDash^* \psi$ and from the definition of D we can see that $(p \parallel q, \pi') \xrightarrow{\tau} (p \parallel q, \pi)$ is the only possible transition between the two computations. Therefore, $(p \parallel q, \pi) \vDash^* \langle \leftarrow \tau \rangle \psi = \langle \leftarrow \alpha \rangle \psi$.

In all cases $(p \parallel q, \pi) \vDash^* \langle \leftarrow \alpha \rangle \psi = \varphi$.

Case $\varphi = \langle \dashrightarrow \rangle \psi$

($\Rightarrow$) First assume $(p \parallel q, \pi) \vDash^* \langle \dashrightarrow \rangle \psi$ and take $(\mu_1, \mu_2) \in D(\pi)$. This means there exists a π' s.t. $(p \parallel q, \pi) \dashrightarrow (p \parallel q, \pi')$ and $(p \parallel q, \pi') \vDash^* \psi$. By definition $(\langle \dashrightarrow \rangle \psi)/(q, \mu_2) = \langle \dashrightarrow \rangle (\psi/(q, \mu_2'))$, where we let $(\mu_1', \mu_2') = (\mu_1(p' \dashrightarrow p'), \mu_2(q' \dashrightarrow q'))$ with $(p' \parallel q') = \text{last}\,(p \parallel q, \pi)$. This is according to the definition of D so $(\mu_1', \mu_2') \in D(\pi')$. Thus, by induction $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$. Since $(p, \mu_1) \dashrightarrow (p, \mu_1')$ we obtain that $(p, \mu_1) \vDash^* \langle \dashrightarrow \rangle (\psi/(q, \mu_2')) = (\langle \dashrightarrow \rangle \psi)/(q, \mu_2)$.

($\Leftarrow$) Assume that $\exists (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* (\langle \dashrightarrow \rangle \psi)/(q, \mu_2)$. We want to show that $(p \parallel q, \pi) \vDash^* \langle \dashrightarrow \rangle \psi$. Let $p' = \text{last}(\mu_1)$ and $q' = \text{last}(\mu_2)$. The formula $\langle \dashrightarrow \rangle \psi)/(q, \mu_2)$ is equal to $\langle \dashrightarrow \rangle (\psi/(q, \mu_2'))$ where $\mu_2' = \mu_2(q' \dashrightarrow q')$. If we let $\pi' = \pi(p' \parallel q' \dashrightarrow p' \parallel q')$ and $\mu_1' = \mu_1(p' \dashrightarrow p')$, then, by definition of D, $(\mu_1', \mu_2') \in D(\pi')$. Observe that $(p, \mu_1) \dashrightarrow (p, \mu_1')$ and that the $\dashrightarrow$ relation is

deterministic. Therefore $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$ for each (μ_1, μ_2). Induction gives that $(p \parallel q, \pi') \vDash^* \psi$. Now it follows trivially that $(p \parallel q, \pi) \vDash^* \langle \dashrightarrow \rangle \psi$ because $(p \parallel q, \pi) \dashrightarrow (p \parallel q, \pi')$.

Case $\varphi = \langle \dashleftarrow \rangle \varphi$

($\Rightarrow$) Assume $(p \parallel q, \pi) \vDash^* \langle \dashleftarrow \rangle \psi$ and take $(\mu_1, \mu_2) \in D(\pi)$. This means that $\pi = \pi'(p' \parallel q' \dashrightarrow p' \parallel q')$ and $(p \parallel q, \pi') \vDash^* \psi$, where $p' \parallel q' = \text{last}(\pi)$. It is obvious, from the definition of D that μ_1 and μ_2 both end with $\dashrightarrow$ since π ends with $\dashrightarrow$. Let π', μ_1', μ_2' be π, μ_1, μ_2 without their last transition respectively (note that our assumption guarantees that they are non-empty). By Lemma 6 we know that $(\mu_1', \mu_2') \in D(\pi')$. Since $(p \parallel q, \pi') \vDash^* \psi$, we have by induction that $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$. Then $(p, \mu_1) \vDash^* \langle \dashleftarrow \rangle (\psi/(q, \mu_2')) = (\langle \dashleftarrow \rangle \psi)/(q, \mu_2)$.

($\Leftarrow$) Now assume $\exists (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* (\langle \dashleftarrow \rangle \psi)/(q, \mu_2)$. Then the last step of μ_2 is $\dashrightarrow$ since otherwise the formula would be equal to $\bot$, which could not be satisfied by (p, μ_1). We see furthermore that the quotiented formula is $\langle \dashleftarrow \rangle (\psi/(q, \mu_2'))$ where μ_2' is again μ_2 without its last step. This means the last step of μ_1 is also $\langle \dashrightarrow \rangle$ (a fact we could also have deduced from the definition of D). Let μ_1' be μ_1 without this step. If we also let π' be π without the last step, then by Lemma 6 we have $(\mu_1', \mu_2') \in D(\pi')$. Since $(p, \mu_1') \vDash^* \psi/(q, \mu_2')$ induction gives that $(p \parallel q, \pi') \vDash^* \psi$. By the definition of D we see that the last step of π can only be $\dashrightarrow$ so $(p \parallel q, \pi) \vDash^* \langle \dashleftarrow \rangle \psi$.

This concludes the analysis of all structural forms for φ. In each case we have shown by structural induction that each direction of the theorem holds. □

Proof of Corollary 1. ($\Rightarrow$) This case follows directly from the theorem.

($\Leftarrow$) Assume that $\forall (\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \vDash^* \varphi/(q, \mu_2)$. Specifically, since there exists at least one decomposition $(\mu_1', \mu_2') \in D(\pi)$, the above holds for that particular decomposition. By the $\Leftarrow$ part of Theorem 1, we thus have that $(p \parallel q, \pi) \vDash^* \varphi$. □

B Proofs for Section 5

Lemma 7. *If paths μ_1 and μ_2 are compatible, then there exists a unique path π such that $(\mu_1, \mu_2) \in D(\pi)$.*

Proof of Lemma 7. The path π is constructed in the obvious way, each transition of it is obtained by composing the processes from the matching transitions of μ_1 and μ_2 with the parallel operator and determining the action according to the rules for that operator. The conditions of Definition 8 ensure that the choice of actions is unambiguous in every case. The rest is easy to check with the definition of D. □

Proof of Theorem 2. The proof follows the lines of the one for Theorem 1. We therefore limit ourselves to considering the case when $\varphi = X$ for $X \in \mathcal{X}$.

Case $\varphi = X \in \mathcal{X}$

($\Rightarrow$) Assume $(p \parallel q, \pi) \vDash_\sigma \varphi$. This means $(p \parallel q, \pi) \in [\![X]\!]\sigma = \sigma(X)$. Now take any $(\mu_1, \mu_2) \in D(\pi)$. By definition of Φ we have that $(p, \mu_1) \in \Phi(\sigma)(X/(q, \mu_2))$, which in turn means that $(p, \mu_1) \in [\![X/(q, \mu_2)]\!]\Phi(\sigma)$, which was to be proved.

($\Leftarrow$) Assume that $(p, \mu_1) \in [\![X/(q, \mu_2)]\!]\sigma' = \sigma'(X/(q, \mu_2))$, for some $(\mu_1, \mu_2) \in D(\pi)$. By the definition of Ψ we obtain directly that $(p \parallel q, \pi) \in \Psi(\sigma')(X)$, which means that $(p \parallel q, \pi) \in [\![X]\!]\Psi(\sigma')$. □

Proof of Lemma 3. Let $\sigma \in \mathcal{E}_\mathcal{X}$. Then for any $X \in \mathcal{X}$

$$\begin{aligned}
& (p \parallel q, \pi) \in (\Psi(\Phi(\sigma))(X) \\
\Leftrightarrow\ & \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \in \Phi(\sigma)(X/(q, \mu_2)) \\
\Leftrightarrow\ & \forall(\mu_1, \mu_2) \in D(\pi) : \exists \pi' : (p \parallel q, \pi') \in \sigma(X) \wedge (\mu_1, \mu_2) \in D(\pi').
\end{aligned}$$

By Lemma 1 we have that in the last line, $\pi = \pi'$, which allows us to continue thus:

$$\begin{aligned}
\Leftrightarrow\ & \forall(\mu_1, \mu_2) \in D(\pi) : (p \parallel q, \pi) \in \sigma(X) \\
\Leftrightarrow\ & (p \parallel q, \pi) \in \sigma(X).
\end{aligned}$$

This shows that the sets $\sigma(X)$ and $(\Psi(\Phi(\sigma)))(X)$ are equal.

Now let $\sigma' \in \mathcal{E}_{\mathcal{X}\times\mathcal{C}}$. For any $X/(q, \mu_2) \in \mathcal{X} \times \mathcal{C}$

$$\begin{aligned}
& (p, \mu_1) \in (\Phi(\Psi(\sigma'))(X/(q, \mu_2)) \\
\Leftrightarrow\ & \exists \pi : (\mu_1, \mu_2) \in D(\pi) \wedge (p \parallel q, \pi) \in \Psi(\sigma')(X) \\
\Leftrightarrow\ & \exists \pi : (\mu_1, \mu_2) \in D(\pi) \wedge (\forall(\mu_1', \mu_2') \in D(\pi) : (p, \mu_1') \in \sigma'(X/(q, \mu_2'))) \\
\Rightarrow\ & \exists \pi : (p, \mu_1) \in \sigma'(X/(q, \mu_2)) \\
\Leftrightarrow\ & (p, \mu_1) \in \sigma'(X/(q, \mu_2)).
\end{aligned}$$

This shows that, for any variable $X/(q, \mu_2)$,

$$(\Phi(\Psi(\sigma')))(X/(q, \mu_2)) \subseteq \sigma'(X/(q, \mu_2)).$$

Now assume $(p, \mu_1) \in \sigma'(X/(q, \mu_2))$, or in other words $(p, \mu_1) \in [\![X/(q, \mu_2)]\!]\sigma'$. By applying the second part of Theorem 2, we obtain

$$(p \parallel q, \pi) \in [\![X]\!]\Psi(\sigma'),$$

where π is the unique path from $p \parallel q$ such that $(\mu_1, \mu_2) \in D(\pi)$. Now we can apply the first part of the theorem in turn, which gives

$$\forall(\mu_1', \mu_2') \in D(\pi) : (p, \mu_1') \in [\![X/(q, \mu_2')]\!]\Phi(\Psi(\sigma')).$$

In particular, this holds for (μ_1, μ_2), i.e. $(p, \mu_1) \in (\Phi(\Psi(\sigma')))(X/(q, \mu_2))$. This shows that

$$\sigma'(X/(q, \mu_2)) \subseteq (\Phi(\Psi(\sigma')))(X/(q, \mu_2)).$$

Since we have shown containment in both directions, it follows that $\sigma' = \Phi(\Psi(\sigma'))$, which concludes the proof. □

Proof of Theorem 3. We limit ourselves to showing that if σ' is a model of $\mathcal{D}'$, then $\Psi(\sigma')$ is a model for the declaration $\mathcal{D}$. The other statement can be proved following similar lines. To this end, assume that σ' is a model of $\mathcal{D}'$. We reason as follows:

$$\begin{aligned}
(p \parallel q, \pi) \models_{\Psi(\sigma')} X &\Leftrightarrow \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \models_{\sigma'} X/(q, \mu_2) \\
&\qquad \text{(Theorem 2)} \\
&\Leftrightarrow \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \models_{\sigma'} \mathcal{D}'(X/(q, \mu_2)) \\
&\qquad (\sigma' \text{ is a model of } \mathcal{D}') \\
&\Leftrightarrow \forall(\mu_1, \mu_2) \in D(\pi) : (p, \mu_1) \models_{\sigma'} \mathcal{D}(X)/(q, \mu_2) \\
&\qquad \text{(Definition of } \mathcal{D}') \\
&\Leftrightarrow (p \parallel q, \pi) \models_{\Psi(\sigma')} X \qquad \text{(Theorem 2).}
\end{aligned}$$

Hence, $\Psi(\sigma')$ is a model for the declaration $\mathcal{D}$, which was to be shown. □