Characteristic Formulae for Fixed-Point Semantics: A General Framework[*]

Luca Aceto Anna Ingolfsdottir Joshua Sack
School of Computer Science, Reykjavik University,
Kringlan 1, IS-103 Reykjavik, Iceland
luca@ru.is, annai@ru.is, joshua.sack@gmail.com

The literature on concurrency theory offers a wealth of examples of characteristic-formula constructions for various behavioural relations over finite labelled transition systems and Kripke structures that are defined in terms of fixed points of suitable functions. Such constructions and their proofs of correctness have been developed independently, but have a common underlying structure. This study provides a general view of characteristic formulae that are expressed in terms of logics with a facility for the recursive definition of formulae. It is shown how several examples of characteristic-formula constructions from the literature can be recovered as instances of the proposed general framework, and how the framework can be used to yield novel constructions.

1 Introduction

Various types of automata are fundamental formalisms for the description of the behaviour of computing systems. For instance, a widely used model of computation is that of *labelled transition systems* (LTSs) [19]. LTSs underlie Plotkin's Structural Operational Semantics [30] and, following Milner's pioneering work on CCS [27], are by now the formalism of choice for describing the semantics of various process description languages.

Since automata like LTSs can be used for describing specifications of process behaviours as well as their implementations, an important ingredient in their theory is a notion of behavioural equivalence or preorder between (states of) LTSs. A behavioural equivalence describes formally when (states of) LTSs afford the same 'observations', in some appropriate technical sense. On the other hand, a behavioural preorder is a possible formal embodiment of the idea that (a state in) an LTS affords at least as many 'observations' as another one. Taking the classic point of view that an implementation correctly implements a specification when each of its observations is allowed by the specification, behavioural preorders may therefore be used to establish the correctness of implementations with respect to their specifications, and to support the stepwise refinement of specifications into implementations.

The lack of consensus on what constitutes an appropriate notion of observable behaviour for reactive systems has led to a large number of proposals for behavioural equivalences and preorders for concurrent processes. In his by now classic paper [14], van Glabbeek presented a taxonomy of extant behavioural preorders and equivalences for processes.

The approach to the specification and verification of reactive systems in which automata like LTSs are used to describe both implementations and specifications of reactive systems is often referred to as *implementation verification* or *equivalence checking*.

[*]The work of the authors has been partially supported by the project "New Developments in Operational Semantics" (nr. 080039021) of the Icelandic Research Fund. Joshua Sack has been further supported by a grant from Reykjavik University's Development Fund.

This is a preliminary version of a paper that will appear in Electronic Proceedings in Theoretical Computer Science.

© Aceto, Ingolfsdottir and Sack
This work is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works License.

An alternative approach to the specification and verification of reactive systems is that of *model checking* [7, 6, 32]. In this approach, automata are still the formalism of choice for the description of the actual behaviour of a concurrent system. However, specifications of the expected behaviour of a system are now expressed using a suitable logic, for instance, a modal or temporal logic [12, 31]. Verifying whether a concurrent process conforms to its specification expressed as a formula in the logic amounts to checking whether the automaton describing the behaviour of the process is a model of the formula.

It is natural to wonder what the connection between these two approaches to the specification and verification of concurrent computation is. A classic, and most satisfying, result in the theory of concurrency is the characterization theorem of bisimulation equivalence [27, 29] in terms of Hennessy-Milner logic (HML) due to Hennessy and Milner [17]. This theorem states that two bisimilar processes satisfy the same formulae in Hennessy-Milner logic, and if the processes satisfy a technical finiteness condition, then they are also bisimilar when they satisfy the same formulae in the logic. This means that, for bisimilarity and HML, the process equivalence induced by the logic coincides with behavioural equivalence, and that, whenever two processes are *not* equivalent, we can always find a formula in HML that witnesses a reason why they are not. This distinguishing formula is useful for debugging purposes, and can be algorithmically constructed for finite processes—see, e.g., [20, 25].

The characterization theorem of Hennessy and Milner is, however, less useful if we are interested in using it directly to establish when two processes are behaviourally equivalent using model checking. Indeed, that theorem seems to indicate that to show that two processes are equivalent we need to check that they satisfy the same formulae expressible in the logic, and there are countably many such formulae, even modulo logical equivalence. Is it possible to find a *single* formula that characterizes the bisimulation equivalence class of a process p—in the sense that any process is bisimilar to p if, and only if, it affords that property? Such a formula, if it exists, is called a *characteristic formula*. When a characteristic formula for a process modulo a given notion of behavioural equivalence or preorder can be algorithmically constructed, implementation verification can be reduced to model checking, and we can translate automata to logic. (An investigation of the model checking problems that can be reduced to implementation verification may, for instance, be found in the paper [4].)

Characteristic formulae provide a very elegant connection between automata and logic, and between implementation verification and model checking. But, can they be constructed for natural, and suitably expressive, automata-based models and known logics of computation? To the best of our knowledge, this natural question was first addressed in the literature on concurrency theory in the paper [16]. In that paper, Graf and Sifakis offered a translation from recursion-free terms of Milner's CCS [27] into formulae of a modal language representing their equivalence class with respect to observational congruence.

Can one characterize the equivalence class of an arbitrary finite process—for instance one described in the regular fragment of CCS—up to bisimilarity using HML? The answer is negative because each formula in that logic can only describe a finite fragment of the initial behaviour of a process—see, for instance, [1] for a textbook presentation. However, as shown in, e.g., [18, 34], adding a facility for the recursive definition of formulae to (variants of) HML yields a logic that is powerful enough to support the construction of characteristic formulae for various types of finite processes modulo notions of behavioural equivalence or preorder.

Following on the work presented in those original references, the literature on concurrency theory offers by now a wealth of examples of characteristic-formula constructions for various behavioural relations over finite labelled transition systems, Kripke structures and timed automata that are defined in terms of fixed points of suitable functions. (See, for instance, the references [2, 5, 8, 13, 21, 23, 28].) Such constructions and their proofs of correctness have been developed independently, but have a common underlying structure. It is therefore natural to ask oneself whether one can provide a general framework within which some of the aforementioned results can be recovered from general principles that isolate the common properties that lie at the heart of all the specific constructions presented in the literature. Not only do such general principles allow us to recover extant constructions in a principled fashion, but they may also yield novel characteristic-formula constructions 'for free'.

In this study, we offer a general view of characteristic formulae that are expressed in terms of logics with a facility for the recursive definition of formulae. The proposed framework applies to behavioural relations that are defined as fixed points of suitable monotonic functions. Examples of such relations are those belonging to the family of bisimulation- and simulation-based semantics. We show that if, in a suitable technical sense defined in Section 3, a recursively defined logical formula expresses the function underlying the definition of a behavioural relation, then the largest interpretation of that formula is exactly the characteristic formula for the derived behavioural relation. (See Theorem 3.3.) Using this result, we are able to recover, as instances of the proposed general framework and essentially for free, several examples of characteristic-formula constructions from the literature. In particular, we focus on simulation [29], bisimulation [27, 29], ready simulation [3, 24], and prebisimulation semantics [26]. In addition, we show that the framework can be used to yield novel constructions. By way of example, we provide characteristic formulae for back and forth bisimilarity, back and forth bisimilarity with indistinguishable states [11] and extended simulation semantics [37].

We trust that the general view of characteristic-formula constructions we provide in this article will offer a framework for the derivation of many more such results and for explaining the reasons underlying the success of extant constructions of this kind in the literature.

Roadmap of the paper The paper is organized as follows. In Section 2 we describe the theoretical background the paper relies on. Section 3 contains the main theorem of the paper whereas Section 4 is devoted to its applications. Finally in Section 5 we give some concluding remarks.

2 Some theoretical background

In this section we provide the theoretical background needed in the paper.

2.1 Fixed points, prefixed points, and postfixed points

Let $\mathbf{X}$ be a set and $\mathscr{F} : \mathscr{P}(\mathbf{X}) \to \mathscr{P}(\mathbf{X})$ be a monotonic function, i.e. a function that preserves the order $\subseteq$. A set $S \subseteq \mathbf{X}$ is a *fixed point* of $\mathscr{F}$ if $\mathscr{F}(S) = S$, a *prefixed point* of $\mathscr{F}$ if $\mathscr{F}(S) \subseteq S$, and a *postfixed point* of $\mathscr{F}$ if $S \subseteq \mathscr{F}(S)$. By Tarski's fixed-point theorem [36], $\mathscr{F}$ has both a unique largest fixed point given by union of all its postfixed points:

$$\mathit{Fix}\mathscr{F} = \bigcup\{S \subseteq \mathbf{X} \mid S \subseteq \mathscr{F}(S)\}$$

and a unique least one given by the intersection of all of its prefixed points:

$$\mathit{fix}\mathscr{F} = \bigcap\{S \subseteq \mathbf{X} \mid \mathscr{F}(S) \subseteq S\}.$$

The largest fixed points of such functions are commonly used as the basis for many co-inductively defined behavioural semantics as we will see later in the paper.

2.2 Logic

We assume a formal specification language or logic $\mathscr{L}(Var)$ defined over a set of variables Var and ranged over by F, possibly with subscripts. We often write $\mathscr{L}$ for $\mathscr{L}(Var)$ if the meaning is clear from the context.

We also involve a function $D : Var \to \mathscr{L}$ called a *declaration*; the declaration can be viewed as providing us with a system of equations over Var that decides the meaning of each variable.

We interpret the language over a given set $\mathbf{P}$. For this purpose we assign to each variable a set of elements in $\mathbf{P}$ for which this variable holds true. To cater for this, we use a function $\sigma : Var \to \mathscr{P}(\mathbf{P})$, called an *environment*. Let $Env(Var)$ be the set of all environments (ranged over by σ) whose domain is Var. Again, we often write Env for $Env(Var)$ if the meaning is clear from the context. Ordered under the pointwise set inclusion, which we again refer to as $\subseteq$, Env is a complete lattice with respect to the induced pointwise $\bigcup$ and $\bigcap$ as the least upper bound and the greatest lower bound respectively.

For each environment $\sigma : Var \to \mathscr{P}(\mathbf{P})$, we let $\models\, \subseteq (Env \times \mathbf{P}) \times \mathscr{L}$ be a relation, with the condition that for each $X \in Var$, we have $(\sigma, p) \models X$ if and only if $p \in \sigma(X)$. A language is monotonic if for every $p \in \mathbf{P}$, formula F and environments σ_1 and σ_2, $(\sigma_1, p) \models F$ implies $(\sigma_2, p) \models F$, whenever $\sigma_1 \subseteq \sigma_2$.

A declaration function D induces a function $[\![D]\!] : Env \to Env$ defined by

$$([\![D]\!]\sigma)(X) = \{p \mid (\sigma, p) \models D(X)\}.$$

Monotonicity of the language guarantees that $[\![D]\!]$ is monotonic. We say that D is monotonic whenever $[\![D]\!]$ is. If $[\![D]\!]$ is monotonic it has both a largest and a least fixed point over Env which we refer to as $\sigma^D_{\max}$ (the largest interpretation of D) and $\sigma^D_{\min}$ (the least interpretation of D) respectively. We also drop the superscript D as the meaning should be clear from the context.

In this study we assume that Var is indexed over some set $\mathbf{P}$, i. e. $Var = \{X_q \mid q \in \mathbf{P}\}$. We say that the largest interpretation of a declaration D gives the characteristic formula for a binary relation S over $\mathbf{P}$ if, for all $p, q \in \mathbf{P}$,

$$p \in \sigma^D_{\max}(X_q) \Leftrightarrow (p, q) \in S.$$

Characteristic formulae given in terms of least interpretations are defined similarly.

3 Expressiveness up-to a relation and characteristic formula

In this section we show how we can derive a characteristic formula for a semantic relation directly from the logical description of the monotonic function that defines the relation. To obtain this, for $S \subseteq \mathbf{P} \times \mathbf{P}$, we define the environment σ_S by

$$\sigma_S(X_q) = \{p \in \mathbf{P} \mid (p, q) \in S\},$$

for each $q \in \mathbf{P}$.

Definition 3.1 *Given a function $\mathscr{F} : \mathscr{P}(\mathbf{P} \times \mathbf{P}) \to \mathscr{P}(\mathbf{P} \times \mathbf{P})$, a declaration $D : Var \to \mathscr{L}$ and a set $S \subseteq \mathbf{P} \times \mathbf{P}$, we say that D* expresses $\mathscr{F}$ up to S *if for any $p, q \in \mathbf{P}$,*

$$(\sigma_S, p) \models D(X_q) \Leftrightarrow\ (p, q) \in \mathscr{F}(S).$$

Next we prove that when D expresses $\mathscr{F}$ up to S then the post- and prefixed points of $[\![D]\!]$ correspond to the post- and prefixed points for $\mathscr{F}$.

Lemma 3.2 *Assume that $\mathscr{F} : \mathscr{P}(\mathbf{P}\times\mathbf{P}) \to \mathscr{P}(\mathbf{P}\times\mathbf{P})$ is a monotonic function and $D : Var \to \mathscr{L}$ is a monotonic declaration such that D expresses $\mathscr{F}$ up to S. Then the following hold:*

$$S \subseteq \mathscr{F}(S) \Leftrightarrow \sigma_S \subseteq [\![D]\!]\sigma_S, \tag{1}$$

and

$$\mathscr{F}(S) \subseteq S \Leftrightarrow [\![D]\!]\sigma_S \subseteq \sigma_S. \tag{2}$$

Proof Assume that $\mathscr{F} : \mathscr{P}(\mathbf{P}\times\mathbf{P}) \to \mathscr{P}(\mathbf{P}\times\mathbf{P})$ is a monotonic function, D is a monotonic declaration, and S is a relation for which D expresses $\mathscr{F}$ up to S. We will only prove (1) as the argument for (2) is similar. Towards proving (1), first assume that $S \subseteq \mathscr{F}(S)$ and that $u \in \sigma_S(X_v)$ for some $u,v \in \mathbf{P}$. By the definition of σ_S, $(u,v) \in S \subseteq \mathscr{F}(S)$. Then, as D expresses $\mathscr{F}$ up to S, $u \in [\![D]\!]\sigma_S(X_v)$.

Next assume that $\sigma_S \subseteq [\![D]\!]\sigma_S$ and that $(u,v) \in S$. This implies that $u \in \sigma_S(X_v) \subseteq [\![D]\!]\sigma_S(X_v)$. As D expresses $\mathscr{F}$ up to S, we have that $(u,v) \in \mathscr{F}(S)$ as desired. □

The following theorem states that if the declaration D expresses $\mathscr{F}$ up to S for every relation S then the largest fixed point of $[\![D]\!]$ characterizes the largest fixed point of $\mathscr{F}$. This means that D, under the largest interpretation, defines the characteristic formula for $Fix\mathscr{F}$. For the sake of completeness, we prove a similar result for the least fixed points although at this point we have not found any concrete applications of that result.

Theorem 3.3 *Assume that $\mathscr{F} : \mathscr{P}(\mathbf{P}\times\mathbf{P}) \to \mathscr{P}(\mathbf{P}\times\mathbf{P})$ is a monotonic function and $D : Var \to \mathscr{L}$ is a monotonic declaration such that D expresses $\mathscr{F}$ up to S for all $S \subseteq \mathbf{P}\times\mathbf{P}$. Then for all $p,q \in \mathbf{P}$,*

1. $(\sigma_{\max}, p) \models X_q \Leftrightarrow (p,q) \in Fix\mathscr{F}$, *and*
2. $(\sigma_{\min}, p) \models X_q \Leftrightarrow (p,q) \in fix\mathscr{F}$.

Proof Assume that $\mathscr{F} : \mathscr{P}(\mathbf{P}\times\mathbf{P}) \to \mathscr{P}(\mathbf{P}\times\mathbf{P})$ is a monotonic function and D is a monotonic declaration that expresses $\mathscr{F}$ up to S for every relation S; that is, for each $S \subseteq \mathbf{P}\times\mathbf{P}$ and $p,q \in \mathbf{P}$,

$$(\sigma_S, p) \models D(X_q) \Leftrightarrow (p,q) \in \mathscr{F}(S).$$

1. We prove that for each $p,q \in \mathbf{P}$,

$$(\sigma_{\max}, p) \models X_q \Leftrightarrow (p,q) \in Fix\mathscr{F},$$

or equivalently that

$$p \in \sigma_{\max}(X_q) \Leftrightarrow (p,q) \in Fix\mathscr{F}.$$

We prove each of the implications separately.

$\Rightarrow$: First define $T = \{(u,v) \mid u \in \sigma_{\max}(X_v)\}$. Then for all $u,v \in \mathbf{P}$,

$$u \in \sigma_T(X_v) \Leftrightarrow (u,v) \in T \Leftrightarrow u \in \sigma_{\max}(X_v).$$

This implies that $\sigma_{\max} = \sigma_T$; in particular σ_T is a fixed point, and hence a postfixed point of $[\![D]\!]$.

Towards proving the statement, assume that $p \in \sigma_{\max}(X_q)$. Then as σ_T is a postfixed point of $[\![D]\!]$ and $\sigma_{\max} = \sigma_T$,

$$p \in \sigma_T(X_q) \subseteq [\![D]\!]\sigma_T(X_q).$$

Since D expresses $\mathscr{F}$ up to T, we may apply (1) in Lemma 3.2 to obtain $T \subseteq \mathscr{F}(T)$. Therefore, as $(p,q) \in T$, we have that $(p,q) \in Fix\mathscr{F}$.

$\Leftarrow$: Assume $(p,q) \in \mathit{Fix}\mathscr{F}$. As $\mathit{Fix}\mathscr{F} = \mathscr{F}(\mathit{Fix}\mathscr{F})$ and D expresses $\mathscr{F}$ up to $\mathit{Fix}\mathscr{F}$, by (1) in Lemma 3.2, we have that

$$\sigma_{\mathit{Fix}\mathscr{F}}(X_q) \subseteq [\![D]\!]\sigma_{\mathit{Fix}\mathscr{F}}(X_q).$$

As $(p,q) \in \mathit{Fix}\mathscr{F}$ implies $p \in \sigma_{\mathit{Fix}\mathscr{F}}(X_q)$, this in turn, implies that $p \in \sigma_{\max}(X_q)$.

2. Now we prove that for each $p,q \in \mathbf{P}$,

$$(\sigma_{\min}, p) \models X_q \Leftrightarrow (p,q) \in \mathit{fix}\mathscr{F},$$

or equivalently that

$$p \in \sigma_{\min}(X_q) \Leftrightarrow (p,q) \in \mathit{fix}\mathscr{F}.$$

We prove each of the implications separately.

$\Rightarrow$: Assume

$$p \in \sigma_{\min}(X_q) = \bigcap_{[\![D]\!]\sigma\subseteq\sigma} \sigma(X_q).$$

In other words, for all σ, $[\![D]\!]\sigma \subseteq \sigma$ implies that $p \in \sigma(X_q)$. We will prove that $(p,q) \in \mathit{fix}\mathscr{F} = \bigcap_{\mathscr{F}(S)\subseteq S} S$ or equivalently that for all S, $\mathscr{F}(S) \subseteq S$ implies that $(p,q) \in S$. Towards proving this, assume that $\mathscr{F}(S) \subseteq S$. We aim at showing that $(p,q) \in S$. Since D expresses $\mathscr{F}$ up to S, by (2) in Lemma 3.2, $[\![D]\!]\sigma_S \subseteq \sigma_S$. By the assumption above, this implies that $p \in \sigma_S(X_q)$, or equivalently $(p,q) \in S$.

$\Leftarrow$: Assume that $(p,q) \in \mathit{fix}\mathscr{F} = \bigcap_{\mathscr{F}(S)\subseteq S} S$, or equivalently that for all $S \subseteq \mathbf{P}$, $\mathscr{F}(S) \subseteq S$ implies that $(p,q) \in S$. We will prove that

$$p \in \sigma_{\min}(X_q) = \bigcap_{[\![D]\!]\sigma\subseteq\sigma} \sigma(X_q).$$

To prove this, it is sufficient to prove that for each environment σ,

$$[\![D]\!]\sigma \subseteq \sigma \text{ implies } p \in \sigma(X_q).$$

Towards proving this, assume that $[\![D]\!]\sigma \subseteq \sigma$. Define

$$T = \{(u,v) \mid u \in \sigma(X_v)\}.$$

Then $\sigma_T = \sigma$ and therefore $[\![D]\!]\sigma_T \subseteq \sigma_T$. Since D expresses $\mathscr{F}$ up to T, by (2) in Lemma 3.2, this implies that $\mathscr{F}(T) \subseteq T$. This in turn implies that $(p,q) \in T$ and therefore that $p \in \sigma_T(X_q) = \sigma(X_q)$ as we wanted to prove. □

We have the following corollary.

Corollary 3.4 *If a declaration D expresses a monotonic function $\mathscr{F}$ (over $\mathscr{P}(\mathbf{P}\times\mathbf{P})$) up to any relation, then the largest interpretation of D gives the characteristic formula for $\mathit{Fix}\mathscr{F}$.*

3.1 Some Observations about Fixed Points

As usual, for a relation $S \subseteq \mathbf{P} \times \mathbf{P}$, we let $S^{-1} = \{(p,q) \mid (q,p) \in S\}$. Furthermore we define $\mathscr{F}^*(S) = (\mathscr{F}(S^{-1}))^{-1}$. We have the following properties:

Lemma 3.5 *The following hold.*

1. *The function $S \mapsto S^{-1}$ is monotonic and bijective.*
2. *If $\mathscr{F}$ is monotonic then $\mathscr{F}^*$ is monotonic.*
3. *$Fix\,\mathscr{F}^* = (Fix\,\mathscr{F})^{-1}$.*

Proof The proofs of $1-2$ follow directly from the definition of S^{-1} and $\mathscr{F}^*$ and we only give the details of the proof of 3. We proceed with this proof as follows:

$$(p,q) \in Fix\,\mathscr{F}^* \Leftrightarrow \exists S.(p,q) \in S \subseteq \mathscr{F}^*(S) \Leftrightarrow$$

$$\exists S.(p,q) \in S \subseteq (\mathscr{F}(S^{-1}))^{-1} \Leftrightarrow$$

$$\exists S.(q,p) \in S^{-1} \subseteq \mathscr{F}(S^{-1}) \Leftrightarrow$$

$$\exists R.(q,p) \in R \subseteq \mathscr{F}(R) \Leftrightarrow$$

$$(q,p) \in Fix\,\mathscr{F} \Leftrightarrow (p,q) \in (Fix\,\mathscr{F})^{-1}.$$

□

4 Applications

In this section we will describe how the general result of Theorem 3.3 can be applied to concrete examples.

We will base these results on variations of a labelled transition system defined as a triple $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$, where

- $\mathbf{P}$ is a set,
- $\mathbf{A}$ is a set of labels and
- $\longrightarrow \subseteq \mathbf{P} \times \mathbf{A} \times \mathbf{P}$ is a transition relation.

As usual, we write $p \xrightarrow{a} p'$ for $(p,a,p') \in \longrightarrow$. We often think of $\mathbf{P}$ as a set of processes, $\mathbf{A}$ as a set of actions, and $p \xrightarrow{a} p'$ as a transition from process p to process p' via action a. We write $p \xrightarrow{a}$ if there exists a q such that $p \xrightarrow{a} q$, and we write $p \not\xrightarrow{a}$ if there is no such q.

All the characteristic-formula constructions we describe in this section apply to labelled transition systems, and variations on that model, that are finite in all of their components.

We will also use variations of the following language. Given a set *Var* of variables and a set $\mathbf{A}$ of actions, we define the language $\mathscr{L}(Var, \mathbf{A})$ to be the standard Hennessy-Milner Logic with recursion (HML)—see, for instance, [22]—, given by the grammar

$$F ::= tt \mid ff \mid X \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle a \rangle F_1 \mid [a]F_1,$$

where $X \in Var$ and $a \in \mathbf{A}$.

Given a labelled transition system $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$, a language $\mathscr{L}(Var, \mathbf{A})$, and an environment $\sigma : Var \to \mathscr{P}(\mathbf{P})$, we define the semantics of the language by a relation $\models$ between $Env(Var) \times \mathbf{P}$ and $\mathscr{L}(Var, \mathbf{A})$, where $(\sigma, p) \models F$ will be read as "F is true at p with respect to σ", and we let $\not\models$ be the complement of $\models$ in $(Env(Var) \times \mathbf{P}) \times \mathscr{L}(Var, \mathbf{A})$. The relation $\models$ is defined by

$$\begin{array}{lll}
(\sigma,p) \models tt & \text{iff} & p \in \mathbf{P} \\
(\sigma,p) \models ff & \text{iff} & (\sigma,p) \not\models tt \\
(\sigma,p) \models X & \text{iff} & p \in \sigma(X) \\
(\sigma,p) \models F_1 \wedge F_2 & \text{iff} & (\sigma,p) \models F_1 \text{ and } (\sigma,p) \models F_2 \\
(\sigma,p) \models F_1 \vee F_2 & \text{iff} & (\sigma,p) \models F_1 \text{ or } (\sigma,p) \models F_2 \\
(\sigma,p) \models \langle a \rangle F_1 & \text{iff} & (\sigma,p') \models F_1 \text{ for some } p' \text{ for which } p \xrightarrow{a} p' \\
(\sigma,p) \models [a] F_1 & \text{iff} & (\sigma,p') \models F_1 \text{ for all } p' \text{ for which } p \xrightarrow{a} p'.
\end{array}$$

One can easily check that the logic is monotonic and therefore the largest and least fixed point constructions, as described in the Section 3, naturally apply.

The following behavioural equivalences are defined as the largest fixed points to monotonic functions.

4.1 Simulation[29]

Given $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$ and $S \subseteq \mathbf{P}$, let $\mathscr{F}_{sim}(S)$ be defined such that

$(p,q) \in \mathscr{F}_{sim}(S)$ iff for every $a \in \mathbf{A}$ and $p' \in \mathbf{P}$,

if $p \xrightarrow{a} p'$ then there exists some $q' \in \mathbf{P}$ such that $q \xrightarrow{a} q'$ and $(p', q') \in S$.

As $\mathscr{F}_{sim}$ is a monotonic function over $\mathscr{P}(\mathbf{P} \times \mathbf{P})$, by Tarski's fixed point theorem, $\mathscr{F}$ has a largest fixed-point, which we denote by $\sqsubseteq_{sim}$.

We now search for characteristic formulas for $\sqsubseteq_{sim}$ in the language $\mathscr{L}(Var, \mathbf{A})$, where $Var = \{X_p \mid p \in \mathbf{P}\}$. First note that, for each $S \subseteq \mathbf{P} \times \mathbf{P}$,

$$(p,q) \in \mathscr{F}_{sim}(S) \Leftrightarrow (\sigma_S, p) \models \bigwedge_{a \in \mathbf{A}} [a] \Big(\bigvee_{q'.q \xrightarrow{a} q'} X_{q'} \Big).$$

Then, by Corollary 3.4, the characteristic formula for $\sqsubseteq_{sim}$ is given by the largest interpretation of the declaration

$$D^{\sqsubseteq}_{sim}(X_q) = \bigwedge_{a \in \mathbf{A}} [a] \Big(\bigvee_{q'.q \xrightarrow{a} q'} X_{q'} \Big).$$

The result above shows that to characterize $\sqsubseteq_{sim}$ we only need the fragment of HML that includes $\vee, \wedge, [a]$ for $a \in \mathbf{A}$, and a set of variables indexed over $\mathbf{P}$.

We now show how we can use Lemma 3.5 to characterize $\sqsupseteq_{sim} = (\sqsubseteq_{sim})^{-1}$, that is, where $p \sqsupseteq_{sim} q$ if and only if $q \sqsubseteq_{sim} p$. The third component of the lemma gives us $\sqsupseteq_{sim} = Fix(\mathscr{F}^*_{sim})$. We eventually aim to characterize simulation equivalence, i. e. the intersection of two preorders , and it is thus helpful to use a new set of variables $Var' = \{Y_q \mid q \in \mathbf{P}\}$ disjoint from Var. We get the following:

$$(p,q) \in \mathscr{F}^*_{sim}(S) \Leftrightarrow (q,p) \in \mathscr{F}(S^{-1}) \Leftrightarrow$$

$$\forall a \in \mathbf{A}, q' \in \mathbf{P}. q \xrightarrow{a} q' \Rightarrow \exists p' \in \mathbf{P}. p \xrightarrow{a} p' \& (q', p') \in S^{-1} \Leftrightarrow$$

$$\forall a \in \mathbf{A}, q' \in \mathbf{P}. q \xrightarrow{a} q' \Rightarrow \exists p' \in \mathbf{P}. p \xrightarrow{a} p' \& (p', q') \in S \Leftrightarrow$$

$$(\sigma_S, p) \models \bigwedge\nolimits_{a, q'.q \xrightarrow{a} q'} \langle a \rangle Y_{q'}.$$

Then, by Corollary 3.4, the characteristic formula for $\sqsupseteq_{sim}$ is given as the largest interpretation of the declaration

$$D^{\sqsupseteq}_{sim}(Y_q) = \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle Y_{q'}.$$

To characterize $\sqsupseteq_{sim}$ we only need the fragment of HML that includes $\wedge$, $\langle a \rangle$ for $a \in \mathbf{A}$, and a set of variables indexed over $\mathbf{P}$.

We can finally use these to define a characteristic formula for simulation equivalence. Define $\sim_{sim}$ such that $p \sim_{sim} q$ iff $p \sqsubseteq_{sim} q$ and $p \sqsupseteq_{sim} q$. Then

$$p \sim_{sim} q \Leftrightarrow (\sigma^{\sqsubseteq}_{\max} \uplus \sigma^{\sqsupseteq}_{\max}, p) \models X_q \wedge Y_q,$$

where $\sigma_1 \uplus \sigma_2$ is defined on $domain(\sigma_1) \cup domain(\sigma_2)$ if $domain(\sigma_1) \cap domain(\sigma_2) = \emptyset$ as

$$\sigma_1 \uplus \sigma_2(z) = \begin{cases} \sigma_1(z) & \text{if } z \in domain(\sigma_1) \\ \sigma_2(z) & \text{otherwise.} \end{cases}.$$

In this case we use the full logic HML.

4.2 Strong bisimulation[29, 27]

Given $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$ and $S \subseteq \mathbf{P}$, let $\mathscr{F}_{bisim}(S)$ be defined such that

$(p,q) \in \mathscr{F}_{bisim}(S)$ iff for every $a \in \mathbf{A}$,

1. if $p \xrightarrow{a} p'$, then there exists $q' \in \mathbf{P}$ such that $q \xrightarrow{a} q'$ and $(p',q') \in S$, and
2. if $q \xrightarrow{a} q'$, then there exists $p' \in \mathbf{P}$ such that $p \xrightarrow{a} p'$ and $(p',q') \in S$.

As $\mathscr{F}_{bisim}$ is monotonic, it has a largest fixed point, which is the seminal notion of bisimulation equivalence that we denote by $\sim_{bisim}$.

As in the case of simulation, the first clause is translated into

$$(\sigma_S, p) \models \bigwedge_{a \in \mathbf{A}} [a] (\bigvee_{q'.q \xrightarrow{a} q'} X_{q'}),$$

and the second one into

$$(\sigma_S, p) \models \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle X_{q'}.$$

Then, by Corollary 3.4, the characteristic formula for $\sim_{bisim}$ is given by the largest interpretation of the declaration.

$$D_{bisim}(X_q) = \bigwedge_{a \in \mathbf{A}} [a] (\bigvee_{q'.q \xrightarrow{a} q'} X_{q'}) \wedge \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle X_{q'}.$$

This is exactly the characteristic formula proposed in [18].

4.3 Ready simulation[3, 24]

Given $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$ and $S \subseteq \mathbf{P}$, let $\mathscr{F}_{RS}(S)$ be defined such that

$(p,q) \in \mathscr{F}_{RS}(S)$ iff for every $a \in \mathbf{A}$ and $q' \in \mathbf{P}$,

1. if $q \xrightarrow{a} q'$, then there exists $p' \in \mathbf{P}$ such that $p \xrightarrow{a} p'$ and $(p',q') \in S$, and

2. if $p \xrightarrow{a}$, then $q \xrightarrow{a}$.

Clearly the second clause can be rewritten as “if $q \stackrel{a}{\not\longrightarrow}$ then $p \stackrel{a}{\not\longrightarrow}$”. Also note that $\mathscr{F}_{RS}$ is monotonic. We denote the largest fixed point of $\mathscr{F}_{RS}$ by $\sqsupseteq_{RS}$.[1]

In HML, $p \stackrel{a}{\not\longrightarrow}$ if and only if $p \models [a]ff$. Therefore, for each $S \subseteq \mathbf{P} \times \mathbf{P}$, we have

$$(p,q) \in \mathscr{F}_{RS}(S) \Leftrightarrow (\sigma_S, p) \models \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle X_q \wedge \bigwedge_{a.q \stackrel{a}{\not\longrightarrow}} [a]ff.$$

By Corollary 3.4, the characteristic formula for $\sqsupseteq_{RS}$ is now given as the largest interpretation of

$$D_{RS}(X_q) = \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle X_q \wedge \bigwedge_{a.q \stackrel{a}{\not\longrightarrow}} [a]ff.$$

4.4 Back and forth bisimulation

So far our examples of applications of Theorem 3.3 have only included known cases from the literature, i. e. where both the semantic relation and its characteristic formula already exist. In this section we will introduce a new semantic equivalence. This is a variant of the back and forth bisimulation equivalence introduced in [10] that assumes several possible past states. The semantics introduced in [10] assumes that the past is unique and consequently the derived equivalence coincides with the standard strong bisimulation equivalence. This is not the case for the multiple possible past semantics considered here. The introduction of this behavioural equivalence serves as a stepping stone towards the one introduced in the subsequent section.

Given $P = (\mathbf{P}, \mathbf{A}, \longrightarrow)$ and $S \subseteq \mathbf{P}$, let $\mathscr{F}_{bfb}(S)$ be defined such that

$(p,q) \in \mathscr{F}_{bfb}(S)$ iff for every $a \in \mathbf{A}$

1. $\forall p' \in \mathbf{P}. p \xrightarrow{a} p' \Rightarrow \exists q' \in \mathbf{P}. q \xrightarrow{a} q'$ and $(p',q') \in S$,
2. $\forall q' \in \mathbf{P}. q \xrightarrow{a} q' \Rightarrow \exists p' \in \mathbf{P}. p \xrightarrow{a} p'$ and $(p',q') \in S$,
3. $\forall p' \in \mathbf{P}. p' \xrightarrow{a} p \Rightarrow \exists q' \in \mathbf{P}. q' \xrightarrow{a} q$ and $(p',q') \in S$ and
4. $\forall q' \in \mathbf{P}. q' \xrightarrow{a} q \Rightarrow \exists p' \in \mathbf{P}. p' \xrightarrow{a} p$ and $(p',q') \in S$.

To express such behaviour in the logical language considered so far, we add two operators $\langle \overline{a} \rangle$ and $[\overline{a}]$ to it for every $a \in \mathbf{A}$. The semantics for these is given by

$$\begin{array}{lll} (\sigma,p) \models \langle \overline{a} \rangle F_1 & \text{iff} & (\sigma,p') \models F_1 \text{ for some } p' \text{ for which } p' \xrightarrow{a} p \text{ and} \\ (\sigma,p) \models [\overline{a}] F_1 & \text{iff} & (\sigma,p') \models F_1 \text{ for all } p' \text{ for which } p' \xrightarrow{a} p. \end{array}$$

Clearly these new operators are monotonic. As in the case for bisimulation equivalence, for each $S \subseteq \mathbf{P} \times \mathbf{P}$, the first two clauses translate into

$$(\sigma_S, p) \models \bigwedge_{a \in \mathbf{A}} [a] \Big(\bigvee_{q'.q \xrightarrow{a} q'} X_{q'} \Big) \wedge \bigwedge_{a,q'.q \xrightarrow{a} q'} \langle a \rangle X_{q'}.$$

The second two clauses involve the new operators:

$$(\sigma_S, p) \models \bigwedge_{a \in \mathbf{A}} [\overline{a}] \Big(\bigvee_{q'.q' \xrightarrow{a} q} X_{q'} \Big) \wedge \bigwedge_{a,q'.q' \xrightarrow{a} q} \langle \overline{a} \rangle X_{q'}.$$

[1] We choose $\sqsupseteq$ rather than the more familiar $\sqsubseteq$ in order to both use the commonly characteristic formula and establish greater uniformity with the notation used elsewhere in the paper.

Then, by Corollary 3.4, the characteristic formula for $\sim_{bfb}$ is given by the largest interpretation of the declaration

$$D_{bfb}(X_q) = \bigwedge_{a\in\mathbf{A}}[a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) \wedge \bigwedge_{a,q'.q\xrightarrow{a}q'} \langle a\rangle X_{q'} \wedge$$
$$\bigwedge_{a\in\mathbf{A}}[\overline{a}](\bigvee_{q'.q'\xrightarrow{a}q} X_{q'}) \wedge \bigwedge_{a,q'.q'\xrightarrow{a}q} \langle \overline{a}\rangle X_{q'}.$$

4.5 Back and forth bisimulation with indistinguishable states[11]

In this section we consider a version of the back and forth bisimulation from the previous section where some of the states are considered indistinguishable by some external agents. For this purpose we augment our notion of labelled transition systems with a set $\mathscr{I}$ of identities (or agents) and a family of equivalence relation $\{\overset{i}{\cdots} \subseteq \mathbf{P}\times\mathbf{P} \mid i\in\mathscr{I}\}$. Such a structure is called an annotated labelled transition system [11] and is written as $(\mathbf{P},\mathbf{A},\xrightarrow{A},\mathscr{I},\cdots)$.

Given such a structure let $\mathscr{F}_{bfbid}(S)$ be defined such that

$(p,q)\in\mathscr{F}_{bfbid}(S)$ iff for every $a\in\mathbf{A}$ and $i\in\mathscr{I}$,

1. $\forall p'\in\mathbf{P}.\,p\xrightarrow{a}p' \Rightarrow \exists q'\in\mathbf{P}.q\xrightarrow{a}q'$ and $(p',q')\in S$,
2. $\forall q'\in\mathbf{P}.\,q\xrightarrow{a}q' \Rightarrow \exists p'\in\mathbf{P}.p\xrightarrow{a}p'$ and $(p',q')\in S$,
3. $\forall p'\in\mathbf{P}.\,p'\xrightarrow{a}p \Rightarrow \exists q'\in\mathbf{P}.q'\xrightarrow{a}q$ and $(p',q')\in S$,
4. $\forall q'\in\mathbf{P}.\,q'\xrightarrow{a}q \Rightarrow \exists p'\in\mathbf{P}.p'\xrightarrow{a}p$ and $(p',q')\in S$,
5. $\forall p'\in\mathbf{P}.\,p\overset{i}{\cdots}p' \Rightarrow \exists q'\in\mathbf{P}.q\overset{i}{\cdots}q'$ and $(p',q')\in S$ and
6. $\forall q'\in\mathbf{P}.\,q\overset{i}{\cdots}q' \Rightarrow \exists p'\in\mathbf{P}.p\overset{i}{\cdots}p'$ and $(p',q')\in S$.

We denote the largest fixed point of $\mathscr{F}_{bfbid}$ by $\sim_{bfbid}$. We use the logical language for back and forth bisimulation from Section 4.4 and add to it the operators $\langle i\rangle$ and $[i]$ for each $i\in\mathscr{I}$. The semantics for these operators is given by

$$\begin{array}{lll}(\sigma,p)\models\langle i\rangle F_1 & \text{iff} & (\sigma,p')\models F_1 \text{ for some } p' \text{ for which } p\overset{i}{\cdots}p' \text{ and}\\ (\sigma,p)\models[i]F_1 & \text{iff} & (\sigma,p')\models F_1 \text{ for all } p' \text{ for which } p\overset{i}{\cdots}p'.\end{array}$$

These new constructions are clearly monotonic. As for the case for back and forth bisimulation, the first four clauses of $\mathscr{F}_{bfbid}(S)$ can be translated into

$$(\sigma_S,p)\models \bigwedge_{a\in\mathbf{A}}[a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) \wedge \bigwedge_{a,q'.q\xrightarrow{a}q'} \langle a\rangle X_{q'} \wedge$$
$$\bigwedge_{a\in\mathbf{A}}[\overline{a}](\bigvee_{q'.q'\xrightarrow{a}q} X_{q'}) \wedge \bigwedge_{a,q'.q'\xrightarrow{a}q} \langle \overline{a}\rangle X_{q'}$$

and the last two clauses into

$$(\sigma_S,p)\models \bigwedge_{i\in\mathscr{I}}[i](\bigvee_{q'.q\overset{i}{\cdots}q'} X_{q'}) \wedge \bigwedge_{i,q'.q\overset{i}{\cdots}q'} \langle i\rangle X_{q'}.$$

Then, by Corollary 3.4, the characteristic formula for $\sim_{bfbid}$ is given by the largest interpretation of the declaration

$$\begin{aligned} D_{bfb}(X_q) = & \bigwedge_{a\in\mathbf{A}}[a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) \wedge \bigwedge_{a,q'.q\xrightarrow{a}q'}\langle a\rangle X_{q'} \wedge \\ & \bigwedge_{a\in\mathbf{A}}[\overline{a}](\bigvee_{q'.q'\xrightarrow{a}q} X_{q'}) \wedge \bigwedge_{a,q'.q'\xrightarrow{a}q}\langle \overline{a}\rangle X_{q'} \wedge \\ & \bigwedge_{i\in\mathscr{I}}[i](\bigvee_{q'.q\overset{i}{\cdots}q'} X_{q'}) \wedge \bigwedge_{i,q'.q\overset{i}{\cdots}q'}\langle i\rangle X_{q'}. \end{aligned}$$

As a consequence of the existence of this characteristic formula, we obtain a behavioural characterization of the equivalence over states in an annotated labelled transition system induced by the epistemic logic studied in [11]. (More precisely, the logic we study in this section may be seen as the positive version of the one studied in [11], where we use the modal operator $[i]$ in lieu of K_i, read "agent i knows", and its dual.) This solves a problem that was left open in the aforementioned reference.

Theorem 4.1 *Let $p,q \in \mathbf{P}$. Then $p \sim_{bfbid} q$ if, and only if, p and q satisfy the same formulae expressible in the logic considered in this section.*

4.6 Prebisimulation[26, 35]

We now extend the original labelled transition system with a convergence predicate $\downarrow$ as found in [35]. We write $p\downarrow$ for $p \in\downarrow$, and we interpret $p\downarrow$ to mean that the process p converges. If $p \not\in\downarrow$, we write $p\uparrow$, and understand it to mean that the process p diverges.

Given $(\mathbf{P},\mathbf{A},\longrightarrow,\downarrow)$ and $S \subseteq \mathbf{P}\times\mathbf{P}$, we define $\mathscr{F}_{prbis}(S)$ such that

$(p,q) \in \mathscr{F}_{prbis}(S)$ iff for every $a \in \mathbf{A}$,

1. for all $q' \in \mathbf{P}$ if $q \xrightarrow{a} q'$ then there exists $p' \in \mathbf{P}$ such that $p \xrightarrow{a} p'$ and $(p',q') \in S$, and
2. if $q\downarrow$, then both of the following hold:
   (a) $p\downarrow$ and
   (b) for all $p' \in \mathbf{P}$, if $p \xrightarrow{a} p'$ then there exists $q' \in \mathbf{P}$, such that $q \xrightarrow{a} q'$ and $(p',q') \in S$.

As $\mathscr{F}_{prbis}$ is monotonic, it has a largest fixed point $\sqsupseteq_{prbis}$,[2] known as the prebisimulation preorder.

To characterize this preorder we use an intuitionistic version HML$_{int}$ of the standard HML. The syntax is the same as before, but the definition for $[a]F$ is now

$$(\sigma,p) \models [a]F \quad \text{iff} \quad p\downarrow \text{ and } (\sigma,p') \models F \text{ for all } p' \text{ for which } p \xrightarrow{a} p'.$$

This implies that the first defining clause is the same as for simulation

$$(\sigma_S,p) \models \bigwedge_{a,q'.q\xrightarrow{a}q'} \langle a\rangle X_{q'},$$

whereas the second one can be expressed as

$$(\sigma_S,p) \models \begin{cases} \bigwedge_{a\in\mathbf{A}}[a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) & \text{if } q\downarrow \\ tt & \text{otherwise.} \end{cases}$$

[2]For a similar reason as for the ready simulation, we use $\sqsupseteq_{prbis}$ rather than $\sqsubseteq_{prbis}$.

This can be written differently as

$$(\sigma_S, p) \models \bigwedge_{a\in\mathbf{A}} [a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) \vee \{tt \mid q\uparrow\},$$

where the notation $\vee\{tt \mid q\uparrow\}$ means that the disjunct tt is present only when $q\uparrow$.

Now, by Corollary 3.4, the characteristic formula for $\sqsupseteq_{prbis}$ is given as the largest interpretation of the declaration

$$D_{prbis}(X_q) = \bigwedge_{a,q'.q\xrightarrow{a}q'} \langle a\rangle X_{q'} \wedge [\bigwedge_{a\in\mathbf{A}} [a](\bigvee_{q'.q\xrightarrow{a}q'} X_{q'}) \vee \{tt \mid q\uparrow\}].$$

4.7 Extended simulation[37]

We now consider labelled transition systems augmented with a preorder relation $\sqsubseteq_{\mathbf{A}}$ over the set $\mathbf{A}$ of labels. Given $(\mathbf{P}, \mathbf{A}, \longrightarrow, \sqsubseteq_{\mathbf{A}})$ and $S \subseteq \mathbf{P}$, we define $\mathscr{F}_{ext}$ such that

$(p,q) \in \mathscr{F}_{ext}(S)$ iff for every $a \in \mathbf{A}$,

if $p \xrightarrow{a} p'$ then there exists $q' \in \mathbf{P}$ and $b \in \mathbf{A}$ such that $a \sqsubseteq_{\mathbf{A}} b$, $q \xrightarrow{b} q'$, and $(p',q') \in S$.

We denote the largest fixed point of $\mathscr{F}_{ext}$ by $\sqsubseteq_{ext}$.

To define the characteristic formula for $\sqsubseteq_{ext}$ we use the standard HML with recursion. First note that for each $S \subseteq \mathbf{P}\times\mathbf{P}$ and $p,q \in \mathbf{P}$

$$(p,q) \in \mathscr{F}_{ext}(S) \Leftrightarrow (\sigma_S, p) \models \bigwedge_{a\in\mathbf{A}} [a](\bigvee_{b.a\sqsubseteq_{\mathbf{A}}b} \bigvee_{q'.q\xrightarrow{b}q'} X_{q'}).$$

By Corollary 3.4, the characteristic formula for $\sqsubseteq_{ext}$ is therefore given as the largest interpretation of

$$D^{\sqsubseteq}_{ext}(X_q) = \bigwedge_{a\in\mathbf{A}} [a](\bigvee_{b.a\sqsubseteq_{\mathbf{A}}b} \bigvee_{q'.q\xrightarrow{b}q'} X_{q'}).$$

As with simulation, we use Lemma 3.5 to characterize $\sqsupseteq_{ext} = (\sqsubseteq_{ext})^{-1}$. We get that the characteristic formula for this preorder is obtained as the largest interpretation of the following declaration:

$$D^{\sqsupseteq}_{ext}(X_q) = \bigwedge_{a\in\mathbf{A}} \bigwedge_{q'.q\xrightarrow{a}q'} \bigvee_{b.a\sqsubseteq_{\mathbf{A}}b} \langle b\rangle X_{q'}.$$

5 Conclusion

This paper provides a general view of characteristic formulae for suitable behavioural relations. The relations of interest are those that can be defined by largest or smallest fixed points of monotonic functions, which can be expressed by declarations over a given language. Theorem 3.3 shows that a declaration that expresses such a function can be viewed as the characteristic formula of either the largest or least fixed point of the function. We have explored a number of applications of this theorem, some in recovering characteristic formulae already discovered, and some being novel constructions. But each of the behavioural relations we consider are largest fixed points of functions, and we hope future work can yield characteristic formulae for interesting least fixed points as well. There are still, however, many other largest fixed points that may be applications of this theorem. These include weak bisimulation equivalence [27], weak bisimulation congruence [27], branching bisimulation equivalence [15], resource bisimulation equivalence [9], and g-bisimulation equivalence [33].

References

[1] Luca Aceto, Anna Ingolfsdottir, Kim G. Larsen & Jiří Srba (2007): *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press.

[2] Luca Aceto, Anna Ingolfsdottir, Mikkel Lykke Pedersen & Jan Poulsen (2000): *Characteristic formulae for timed automata*. *RAIRO, Theoretical Informatics and Applications* 34(6), pp. 565–584. Available at `http://dx.doi.org/10.1051/ita:2000131`.

[3] Bard Bloom, Sorin Istrail & Albert Meyer (1995): *Bisimulation can't be Traced*. *Journal of the ACM* 42(1), pp. 232–268.

[4] Gérard Boudol & Kim G. Larsen (1992): *Graphical versus logical specifications*. *Theoretical Computer Science* 106(1), pp. 3–20.

[5] M.C. Browne, E.M. Clarke & O. Grümberg (1988): *Characterizing finite Kripke structures in propositional temporal logic*. *Theoretical Computer Science* 59(1,2), pp. 115–131.

[6] Ed Clarke, Orna Gruemberg & Doron Peled (1999): *Model Checking*. MIT Press.

[7] E.M. Clarke & E.A. Emerson (1981): *Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic*. In: D. Kozen, editor: *Proceedings of the Workshop on Logics of Programs*, *Lecture Notes in Computer Science* 131. Springer-Verlag, pp. 52–71.

[8] Rance Cleaveland & Bernhard Steffen (1991): *Computing Behavioural Relations, Logically*. In: J. Leach Albert, B. Monien & M. Rodríguez, editors: *Proceedings 18th ICALP*, Madrid, *Lecture Notes in Computer Science* 510. Springer-Verlag, pp. 127–138.

[9] Flavio Corradini, Rocco De Nicola & Anna Labella (1999): *Graded Modalities and Resource Bisimulation*. In: C. Pandu Rangan, Venkatesh Raman & Ramaswamy Ramanujam, editors: *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, *Lecture Notes in Computer Science* 1738. Springer-Verlag, pp. 381–393. Available at `http://link.springer.de/link/service/series/0558/bibs/1738/17380381.htm`.

[10] Rocco De Nicola, Ugo Montanari & Frits Vaandrager (1990): *Back and forth bisimulations*. In: *CONCUR' 90 (Amsterdam, 1990)*, *Lecture Notes in Comput. Sci.* 458. Springer, Berlin, pp. 152–165.

[11] Francien Dechesne, MohammadReza Mousavi & Simona Orzan (2007): *Operational and Epistemic Approaches to Protocol Anlaysis: Bridging the Gap*. In: *Proceedings of the 14th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'07)*, *Lecture Notes in Artificial Intelligence* 4790. Springer-Verlag, pp. 226–241.

[12] E. Allen Emerson (1990): *Temporal and modal logic*. In: *Handbook of Theoretical Computer Science, Vol. B*. Elsevier, Amsterdam, pp. 995–1072.

[13] Harald Fecher & Martin Steffen (2005): *Characteristic μ-Calculus Formula for an Underspecified Transition System*. In: *EXPRESS'04*, *Electronic Notes in Theoretical Computer Science* 128. Elsevier Science Publishers, pp. 103–116. Available at `http://www.informatik.uni-kiel.de/~hf/papers/Fecher04express.pdf`.

[14] R. van Glabbeek (2001): *The linear time–branching time spectrum. I. The semantics of concrete, sequential processes*. In: Jan Bergstra, Alban Ponse & Scott A. Smolka, editors: *Handbook of Process Algebra*. Elsevier, pp. 3–99.

[15] R. van Glabbeek & W.P. Weijland (1996): *Branching Time and Abstraction in Bisimulation Semantics*. *Journal of the ACM* 43(3), pp. 555–600.

[16] S. Graf & J. Sifakis (1986): *A Modal Characterization of Observational Congruence on Finite Terms of CCS*. *Information and Control* 68(1–3), pp. 125–145.

[17] M. Hennessy & R. Milner (1985): *Algebraic laws for nondeterminism and concurrency*. *Journal of the ACM* 32(1), pp. 137–161.

[18] Anna Ingolfsdottir, Jens Christian Godskesen & Michael Zeeberg (1987): *Fra Hennessy-Milner Logik til CCS-Processer*. Master's thesis, Department of Computer Science, Aalborg University. In Danish.

[19] R.M. Keller (1976): *Formal verification of parallel programs*. *Communications of the ACM* 19(7), pp.

371–384.

[20] H. Korver (1992): *Computing Distinguishing Formulas for Branching Bisimulation*. In: K.G. Larsen & A. Skou, editors: *Proceedings of the Third Workshop on Computer Aided Verification*, Aalborg, Denmark, July 1991, *Lecture Notes in Computer Science* 575. Springer-Verlag, pp. 13–23.

[21] F. Laroussinie, K. G. Larsen & C. Weise (1995): *From Timed Automata to Logic - and Back*. In: Jirí Wiedermann & Petr Hájek, editors: *Mathematical Foundations of Computer Science 1995, 20th International Symposium*, *Lecture Notes in Computer Science* 969. Springer, Prague, Czech Republic, pp. 529–539.

[22] Kim Guldstrand Larsen (1990): *Proof Systems for Satisfiability in Hennessy–Milner Logic with Recursion*. *Theoretical Computer Science* 72(2–3), pp. 265–288.

[23] Kim Guldstrand Larsen & A. Skou (1992): *Compositional Verification of Probabilistic Processes*. In: Rance Cleaveland, editor: *Proceedings CONCUR 92*, Stony Brook, NY, USA, *Lecture Notes in Computer Science* 630. Springer-Verlag, pp. 456–471.

[24] Kim Gulstrand Larsen & A. Skou (1991): *Bisimulation through Probabilistic Testing*. *Information and Computation* 94(1), pp. 1–28.

[25] Tiziana Margaria & Bernhard Steffen (1993): *Distinguishing Formulas for Free*. In: *Proc. EDAC–EUROASIC'93: IEEE European Design Automation Conference, Paris (France)*. IEEE Computer Society Press.

[26] R. Milner (1981): *A modal characterisation of observable machine behaviour*. In: E. Astesiano & C. Böhm, editors: *CAAP '81: Trees in Algebra and Programming, 6th Colloquium*, *Lecture Notes in Computer Science* 112. Springer-Verlag, pp. 25–34.

[27] R. Milner (1989): *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs.

[28] Markus Müller-Olm (1998): *Derivation of Characteristic Formulae*. In: *MFCS'98 Workshop on Concurrency (Brno, 1998)*, *Electron. Notes Theor. Comput. Sci.* 18. Elsevier, Amsterdam, p. 12 pp. (electronic).

[29] D. Park (1981): *Concurrency and automata on infinite sequences*. In: P. Deussen, editor: *5th GI Conference*, Karlsruhe, Germany, *Lecture Notes in Computer Science* 104. Springer-Verlag, pp. 167–183.

[30] Gordon D. Plotkin (2004): *A Structural Approach to Operational Semantics*. *Journal of Logic and Algebraic Programming* 60–61, pp. 17–139.

[31] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: *Proceedings 18th Annual Symposium on Foundations of Computer Science*. IEEE, pp. 46–57.

[32] J. P. Queille & J. Sifakis (1981): *Specification and Verification of Concurrent Systems in Cesar*. In: *Proceedings of the 5th International Symposium on Programming*, *Lecture Notes in Computer Science* 137. Springer-Verlag, pp. 337–351.

[33] M. de Rijke (2000): *A Note on Graded Modal Logic*. *Studia Logica* 64(2), pp. 271–283.

[34] Bernhard Steffen & Anna Ingolfsdottir (1994): *Characteristic Formulae for Processes with Divergence*. *Information and Computation* 110(1), pp. 149–163.

[35] Colin Stirling (1987): *Modal logics for communicating systems*. *Theoret. Comput. Sci.* 49(2-3), pp. 311–347. Twelfth international colloquium on automata, languages and programming (Nafplion, 1985).

[36] A. Tarski (1955): *A Lattice-Theoretical Fixpoint Theorem and its Applications*. *Pacific Journal of Mathematics* 5, pp. 285–309.

[37] B. Thomsen (1987): *An extended bisimulation induced by a preorder on actions*. Master's thesis, Aalborg University Centre.