Weak bisimilarity coalgebraically

Andrei Popescu

University of Illinois at Urbana-Champaign and
Institute of Mathematics Simion Stoilow of the Romanian Academy

Abstract. We argue that weak bisimilarity of processes can be conveniently captured in a semantic domain by a combination of traces and coalgebraic finality, in such a way that important process algebra aspects such as parallel composition and recursion can be represented compositionally. We illustrate the usefulness of our approach by providing a fully-abstract denotational semantics for CCS under weak bisimilarity.

1 Introduction

Weak bisimilarity (henceforth referred to simply as *bisimilarity*) is one of the most interesting equivalences defined on nondeterministic interactive processes. A nondeterministic process p may take one of several actions a_i and transit to a process p_i; among the possible actions, there is the *silent*, *unobservable* action, τ. Intuitively, a process may take a τ action at any time without being noticed, while any other action is *observable*. Now, the *(behavioral) identity* of two processes p_1 and p_2 can be determined by the following two clauses (and their symmetrics): **(i)** if p_1 may do something internally (i.e., take zero or more τ actions) becoming p_1', then p_2 may as well do something internally becoming p_2', behaviorally identical to p_1'; **(ii)** if p_1 may do something internally, then take an observable action a, then again do something internally eventually becoming p_1', then p_2 may as well go through a similar series of steps (consisting of action a possibly succeeded and/or preceded by some silent steps) eventually becoming p_2', behaviorally identical to p_1'. The *strong bisimilarity* of processes is a useful, but often too crude, approximation of bisimilarity, obtained by assuming the internal action τ observable.

The typical route for introducing processes is through some syntactic constructs building terms whose behavior is given by an SOS-specified labeled transition system. The identity of a process can then be taken to be the bisimilarity class of a term. A more direct, and often more insightful way of establishing the identity of the described processes is to show that process terms really describe processes, in other words, to assign a denotation from a *domain of processes* (or *behaviors*) to each term. The agreement between bisimilarity defined operationally and the kernel of the denotation map is referred to as *full abstraction*.

Bisimilarity, although extensively studied in various operational settings [14, 15, 20], has not benefited yet, in our opinion, from a suitable compositional denotational-semantics account within process algebra. (Although a certain ad hoc compositionality for CCS and CSP has been achieved in [2] and could be achieved along the lines of [18], and non-compositional denotational models for CCS, CSP and π under weak bisimilarity have been sketched in [12, 13] – see

also Section 4.) To the contrary, *strong* bisimilarity, having a simpler structure, has received compositional fully-abstract semantics, both domain-theoretic [1, 10, 22] and coalgebraic [18, 24], based on hypotheses that hold (or merely using techniques that work) for many process calculi in current use. The same holds true for *may/must testing equivalence*, also simpler than bisimilarity but in a different way (namely, in that establishing it does not involve any alternating game as for strong bisimilarity and bisimilarity) [6, 11, 3].

The difficulty with assigning compositional denotational semantics for bisimilarity seems to emerge from the fundamental divorce between two features:

(I) the traditional one-action-depth, "branching-time" presentation of a process calculus;

(II) the "linear-time" consideration of τ^*-sequences in the definition of bisimilarity.

Considering (I), one is tempted to transform right away the conditional rules from the SOS presentation of the system into corresponding corecursive or fixpoint recursive definitions, leading to problems with (II).

The solution proposed by this paper comes from resisting this temptation, and acknowledging that, due to the possibility of melting observable actions into silent actions via communication, *in order to handle what happens arbitrarily τ-deep into processes, one also needs to deal with what happens arbitrarily deep along any sequence of actions.* This might suggest to abandon coalgebraic semantics altogether and go after some form of pure trace semantics,[1] but this would be a mistake, since the very nature of bisimilarity is coalgebraic – the infinite game that defines bisimilaritiy is a journey into processes with infinitely often occurring stops and restarts. Instead, we shall combine traces with coalgebra, identifying a process with all pairs (traceOfActions,continuationProcess) such that that trace of actions is possible and leads to that continuation.

Technically, we shall regard bisimilarity as strong bisimilarity where the "actions" are now sequences of *observable* actions, and where a chain of zero or more τ actions is represented by the empty sequence ϵ. The monoidal properties of the set of action sequences shall be reflected in modal axioms for the final coalgebraic semantic domain. Of course, defining semantic operations within such an approach requires a preliminary combinatorial study of sequences/traces of actions, that need to be shuffled in consistent ways. As it happens, paying a priori attention to traces has its rewards – operations on processes like parallel composition and iteration, main characters in process algebra, receive concise, uniform and conceptually clear definitions.

After we discuss the above constructions in an abstract semantic domain (without reference to syntax), we illustrate the usefulness of our approach by defining a novel denotational semantics for Milner's Calculus of Communicating Systems (CCS) [14]. The semantics will be both *compositional* and *fully abstract*.

[1] In this paper "coalgebraic semantics" will mean: semantics involving a domain which is a final coalgebra of some kind, and thus it is inhabited by items having a relevant *branching structure*. We contrast this with a (pure) *trace semantics*, where the branching structure is ignored.

Here is how the remaining of the paper is structured. The rest of this section is dedicated to general mathematical preliminaries and conventions. Section 2 discusses, in coalgebraic terms, semantic models for weak bisimilarity. Section 3 shows that our approach yields naturally a fully-abstract semantics for CCS under weak bisimilarity. Section 4 draws conclusions and discusses related work.

Preliminaries and conventions. When we introduce a metavariable (such as x) to range over a certain domain, we implicitly assume that this is also the case for its version decorated with accents and subscripts (such as x_1, x'). We use λ-abstractions informally, at the meta-level.

Throughout our exposition, we shall make the standard distinction between classes and sets, and we shall occasionally refer to collections and families of classes – such references could be avoided in well-known ways, or alternatively we could consider a three-level tower of Grothendieck universes, where we could place our sets, classes and collections. Thanks to their standardness w.r.t. our constructions, we shall not worry about foundational issues in this paper, beyond insisting that for a class C, $\mathcal{P}(C)$ denotes the class of *subsets* (and not subclasses) of C.

By "relation" we mean, by default, "binary relation". For a relation $R \subseteq A \times B$, $R^\smile \subseteq B \times A$ is its converse. Given a function $f : A \to B$ and $A' \subseteq A$, $Im(f)$ is the image of f and $f|_{A'}$ is the restriction of f to A'. We usually denote function application by juxtaposition, as in $f\,x$, but sometimes (especially when dealing with *fixed* operators) also employ the parenthesized notation $f(x)$. Given $f : A \to B$, $a \in A$ and $b \in B$, $f[a \leftarrow b] : A \to B$ is defined by $(f[a \leftarrow b])a' = b$ if $a' = a$ and $= f\,a'$ otherwise. A^* is the set of finite sequences/traces of items in A, i.e., the set of words over the alphabet A. $\#$ denotes, in an infixed manner, word concatenation, and ϵ the empty word. (I.e., $(A^*, \#, \epsilon)$ is the monoid freely generated by A.) Thus, for two words $w_1, w_2 \in A^*$ (unless otherwise stated), $w_1 \# w_2$, and *not* the simple juxtaposition $w_1 w_2$, denotes their concatenation. However, given the elements $a_1, \ldots, a_n \in A$, we write $a_1 \ldots a_n$ for the word (sequence) built with these letters (in this order). A^+ denotes $A^* \setminus \{\epsilon\}$. $\mathbb{N}$ is the set of natural numbers. For $m, n \in \mathbb{N}$, $\overline{m,n}$ is $\{i.\, m \leq i \leq n\}$. Unless otherwise stated, m, n, i, j, k range over $\mathbb{N}$. In proofs, "IH" means "the inductive hypothesis".

2 Semantic domain for weak bisimilarity

In this section, we present a semantic domain for weak bisimilarity based on traces of actions, and show its connection with, and its advantage over a more standard domain based on single actions.

Cls denotes the hyper-category of classes and functions between classes. In what follows, we shall employ basic concepts and results about coalgebras [19]. Given a functor $F : \mathit{Cls} \to \mathit{Cls}$, an F-coalgebra is a pair $(D, \delta : D \to F\,D)$. A *morphism* between two F-coalgebras (D, δ) and (D', δ') is a map $f : D \to D'$ such that $(F\,f) \circ \delta = \delta' \circ f$. A *stable part* of an F-coalgebra (D, δ) is a subclass $X \subseteq D$ such that $\forall x \in X.\, \delta\,x \in F\,X$; a stable part X yields a *subcoalgebra*

$(X, \delta|_X)$. Given a fixed set Z, the functor $\mathcal{P}(Z \times \_)$ on Cls maps each X to $\mathcal{P}(Z \times X)$ and each $f : X \to Y$ to $\lambda K.\{(z, f\,x).(z, x) \in K\}$. Fix Act, a set of *actions*. We only consider coalgebras for $\mathcal{P}((Act \cup \{\epsilon\}) \times \_)$ and $\mathcal{P}(Act^* \times \_)$, which we call *one-step*, and *multi-step coalgebras*, respectively. (We have these functors act on *classes* rather than sets in order to ensure the existence of their final coalgebras – see below. An alternative would have been to replace $\mathcal{P}(X)$ with $\mathcal{P}_k(X)$, the set of subsets of X of cardinality less than k, for a large enough cardinal k (such as $2^{\aleph_0}$) – we prefer our current solution since it does not commit to any arbitrary choice.)

Domains of processes under strong bisimilarity are typically modeled as final *one-step* coalgebras. However, if the desired process identity is bisimilarity, then (also having in mind that operations like parallel composition need to take a deeper, multi-step look into the argument processes) it is more natural to consider a suitable *multi-step* coalgebra as the domain. But also we would like to keep in sight that bisimilarity is a weakening of strong bisimilarity by internalizing τ, in particular, to be able to infer bisimilarity from strong bisimilarity without any detour, and also to infer bisimilarity as usual, by showing how to simulate single steps only (by multi-steps). These lead to the following constructions.

Given a multi-step coalgebra (D, δ), an element $d \in D$ is said to be:

- *reflexive*, if $(\epsilon, d) \in \delta\, d$;
- *transitive*, if $\forall w, w', d', d''.\ (w, d') \in \delta\, d \wedge (w', d'') \in \delta\, d' \Rightarrow (w\#w', d'') \in \delta\, d$;
- *prefix-closed*, if $\forall w, w', d''.\ (w\#w', d'') \in \delta\, d \Rightarrow (\exists d'.\ (w, d') \in \delta\, d \wedge (w', d'') \in \delta\, d')$;
- *monoidal*, if it is reflexive, transitive and prefix-closed.

(D, δ) is said to be *monoidal* if all elements of D are monoidal.

Let $(Preproc, unf : Preproc \to \mathcal{P}(Act^* \times Preproc))$ be the final multi-step coalgebra. We call the elements of $Preproc$ *preprocesses*, and unf the *unfolding map*. Let $Proc$, the class of *processes*, be the stable part of $(Preproc, unf)$ cogenerated by the class of all monoidal preprocesses. This notion of process encompasses two ideas. First, processes have a *linear-time* structure which is compatible with the monoid of action sequences (via reflexivity and transitivity) and has no discontinuities (via prefix-closeness). Second, processes have the above linear-time properties preserved by the *branching-time*, coalgebraic structure – one should think of a process as an *hereditarily* monoidal preprocess, that is, a preprocess π such that π is monoidal and the preprocesses from all its arbitrarily deep unfoldings are so.

The properties of the linear-time structure, making sense for any transition system labeled with sequences of actions (i.e., for any multi-step coalgebra), are the essential features of weak bisimilarity. Our choice to impose these properties hereditarily deep for elements in the final multi-step coalgebra makes $(Proc, unf : Proc \to \mathcal{P}(Act^* \times Proc))$ (where unf here denotes the restriction of $unf : Preproc \to \mathcal{P}(Act^* \times Preproc)$) the *final monoidal multi-step coalgebra,* featuring the following *corecursive definition* principle: to define a (parameterized) operation $f : Param \times Proc^n \to Proc$, it suffices to organize $Param \times Proc^n$ as a *monoidal* multi-step coalgebra by defining δ :

$Param \times Proc^n \to \mathcal{P}(Act^* \times (Param \times Proc^n))$ (then take f to be the unique morphism between $(Param \times Proc^n, \delta)$ and $(Proc, unf)$).

Moreover, the fact that $(Proc, unf)$ is *simple* (being a subcoalgebra of the (absolutely) final coalgebra), yields standardly a (coinductive) proof principle: Assume $\theta \subseteq Proc \times Proc$ is a strong bisimulation on $Proc$ (regarded as an Act^*-labeled transition system), in that the following hold for all $(\pi, \pi') \in \theta$:

- $\forall (w, \pi'') \in unf(\pi).\, \exists \pi'''.\, (w, \pi''') \in unf(\pi') \wedge (\pi'', \pi''') \in \theta$;
- $\forall (w, \pi'') \in unf(\pi').\, \exists \pi'''.\, (w, \pi''') \in unf(\pi) \wedge (\pi''', \pi'') \in \theta$.

Then $\pi = \pi'$ for all $(\pi, \pi') \in \theta$.

Thanks to the affinity between $Proc$ and the monoidal structure of Act^*, we also have a simpler (but equally powerful) proof principle which reflects more closely the traditional way of dealing with weak bisimilarity, by showing how to simulate single actions only:

Let $\theta \subseteq Proc \times Proc$ be such that the following hold for all $(\pi, \pi') \in \theta$:

- $\forall (w, \pi'') \in unf(\pi).\, |w| \in \{0, 1\} \Rightarrow (\exists \pi'''.\, (w, \pi''') \in unf(\pi') \wedge (\pi'', \pi''') \in \theta)$;
- $\forall (w, \pi'') \in unf(\pi').\, |w| \in \{0, 1\} \Rightarrow (\exists \pi'''.\, (w, \pi''') \in unf(\pi) \wedge (\pi''', \pi'') \in \theta)$.

Then $\pi = \pi'$ for all $(\pi, \pi') \in \theta$.

The latter proof principle may appear, at first, as if employing *strong* bisimulation (w.r.t. *single* actions) – but remember that processes absorb the monoidal properties of action sequences; in particular: ϵ is identified with (any sequence in the language) ϵ^*, thus meaning "zero or more silent steps"; the single action a is identified with $\epsilon^* \# a \# \epsilon^*$, meaning "$a$ preceded and succeeded by zero or more silent steps". Thus, *weak* bisimulation is what we really deal with here, strong bisimulation being only a particular case.

The fact that each process is uniquely identified by its behavior w.r.t. sequences of length 0 or 1 (i.e., elements of $Act \cup \{\epsilon\}$) suggests a more compact representation of the domain of processes as a one-step coalgebra. As already mentioned, the choice of the (absolutely) final one-step coalgebra as the semantic domain for *strong* bisimilarity typically already yields the desired properties (in particular, full abstraction [18]). However, here, since we are after bismilarity, we shall require that processes, even in this more compact representation with single steps, retain the affinity with the monoid of action sequences – this means here that zero or more ϵ-steps can always be appended and/or prepended "silently", yielding a property similar to monoidality that we shall call ϵ-monoidality. (Although the constructions below are not needed later in the paper, they are useful for placing our domain in a context that clarifies its connection with the traditional view on bisimilarity and its benefit w.r.t. compositionality.)

Let $(Cpreproc, unfc)$ be the final one-step coalgebra. We call the elements of *Cpreproc compact (representations of) preprocesses*. Given a one-step coalgebra (D, δ), an element $d \in D$ is said to be: *ϵ-reflexive*, if $(\epsilon, d) \in \delta\, d$; *$\epsilon$-transitive*, if $\forall d', d'' \in D.\ (\epsilon, d') \in \delta\, d \wedge (\epsilon, d'') \in \delta\, d' \Rightarrow (\epsilon, d'') \in \delta\, d$; *$\epsilon$-loud-transitive*, if $\forall d', d'', d''' \in D. \forall a \in Act.\ (\epsilon, d') \in \delta\, d \wedge (a, d'') \in \delta\, d' \wedge (\epsilon, d''') \in \delta\, d'' \Rightarrow (a, d''') \in \delta\, d$; *$\epsilon$-monoidal*, if it is ϵ-reflexive, ϵ-transitive and ϵ-loud-transitive. (D, δ) is

said to be ϵ-*monoidal* if all elements of D are ϵ-monoidal. (The ϵ-monoidal one-step coalgebras are essentially Aczel's τ-colagebras [2].) Let *Cproc*, the class of *compact (representations of) processes*, be the stable part of the one-step coalgebra $(Cpreproc, unfc)$ cogenerated by the class of all ϵ-monoidal compact preprocesses. Then $(Cproc, unfc : Cproc \to \mathcal{P}((Act \cup \{\epsilon\}) \times Cproc))$ is the final ϵ-monoidal one-step coalgebra.

Next, we write π for processes and σ for compact processes. Using the finality of *Cproc* and *Proc*, we define two maps, $pack : Proc \to Cproc$ and $unpack : Cproc \to Proc$, for moving back and forth between a process and its compact representation:

- $unfc\,(pack(\pi)) = \{(w, pack(\pi')).\, (w, \pi') \in \pi \wedge w \in Act \cup \{\epsilon\}\}$;
- $unf(unpack(\sigma)) = \{(w_1 \# \ldots \# w_n, unf(unpack(\sigma'))).\, n \in I\!N \wedge (\forall i \in \overline{1,n}.\, w_i \in Act \cup \{\epsilon\}) \wedge (\exists \sigma_1, \ldots, \sigma_{n+1}.\, \sigma_1 = \sigma \wedge \sigma_{n+1} = \sigma' \wedge (\forall i \in \overline{1,n}.\, (w_i, \sigma_{i+1}) \in \sigma_i))\}$.

Thus, $pack(\pi)$ retains from π (and from all its (arbitrarily deep) continuations) only the one-step continuations, while $unpack\,\sigma$ expands σ (and all its continuations) along all possible sequences of steps. *pack* and *unpack* are mutually inverse bijections.

Compact processes are coalgebraically only one-step deep, hence correspond more directly to the traditional operational semantics presentation of process calculi. However, in our context of (weak) bisimilarity, these compact one-step representations have a salient disadvantage compared to (multi-step) processes: crucial operations in process calculi, such as parallel composition and iteration, are not definable purely coalgebraically[2] on compact processes, the one-step coalgebra falling short on providing means to describe the composite process.

To illustrate this point, assume that Act is endowed with a bijection $^- : Act \to Act$ which is involutive (in that $\overline{\overline{a}} = a$ for all $a \in Act$), and say we would like to define CCS-like parallel composition on processes (thus, we assume that a and $\overline{a}$ are to be interpreted as complementary actions, whose synchronization yields a silent action). The main task in front of us is to show how sequences of actions interact, possibly nondeterministically. Knowing how *single actions* interact, and assuming an *interleaving semantics*, we obtain the following recursive definition of the *parallel composition* (or *synchronized shuffle*) $| : Act^* \times Act^* \to \mathcal{P}(Act^*)$:

- $\epsilon|\epsilon = \{\epsilon\}$;
- $(a\#w_1)|(b\#w_2) = a\#(w_1|(b\#w_2)) \cup b\#((a\#w_1)|w_2)$, if $\overline{a} \neq b$;
- $(a\#w_1)|(b\#w_2) = a\#(w_1|(b\#w_2)) \cup b\#((a\#w_1)|w_2) \cup w_1|w_2$, if $\overline{a} = b$.

(Above, we overloaded $\#$ in the usual fashion, to denote both $\# : Act^* \times Act^* \to Act^*$ and its componentwise extension to $Act^* \times \mathcal{P}(Act^*) \to \mathcal{P}(Act^*)$, given by $w \,\#\, S = \{w \,\#\, w'.\, w' \in S\}$.)

Now, parallel composition of processes, $| : Proc \times Proc \to Proc$, simply follows coinductively the interaction law prescribed by action sequence composition:

[2] In the sense that a definition based entirely on coalgebraic finality is not available.

$unf(\pi_1|\pi_2) = \{(w, \pi_1'|\pi_2') : \exists w_1, w_2.\ w \in w_1|w_2 \ \wedge \ (w_1, \pi_1') \in unf(\pi_1) \ \wedge \ (w_2, \pi_2') \in unf(\pi_2)\}$.

Note the separation of concerns in our definition of $|$: first we dealt with action sequence combinatorics, so that afterwards the definition of parallel composition of processes was stated *purely coalgebraically*, composing together the continuations of the components to yield the continuations of the composite. On the other hand, if trying to define parallel composition on *compact processes*, one finds the coalgebraic one-step depth of the latter insufficient for describing even the one-step behavior of the composite – this is because a step of the composite $\sigma_1|\sigma_2$ may result from an *arbitrarily long* sequence of interactions between steps taken by continuations of σ_1 and σ_2. In fact, any reasonable definition of parallel composition on *Cproc* would essentially appeal to our multi-step domain *Proc*, unpacking the components σ_1 and σ_2, composing these unpacked versions in *Proc*, and then packing the result (i.e., the operation on *Cproc* would be essentially $pack((unpack\ \sigma_1)|(unpack\ \sigma_2))$). This suggests that *Proc*, and not *Cproc*, is the fundamental domain for compositional weak bisimilarity.

One may argue that, via the bijections *pack* and *unpack*, *Proc* and *Cproc* are really the same domain, so that considering one or the other are two sides of the same coin. However, *Proc* and *Cproc* take nevertheless two *distinct views* to processes, with the multi-step view brought by *Proc*, as explained above, allowing for a cleaner compositionality and separation of concerns when defining process composition. When the effect of actions becomes more complex, with possible changes of the communication topology, the multi-step view brings an insight into the semantics of a calculus under bisimilarity which is essential for apprehending the "right" semantic operations for full abstraction – this is illustrated in [17], where we define a fully-abstract semantics for the π-calculus under weak bisimilarity. A simpler illustration of our approach is presented in the next section, where we give a novel semantics for CCS which is both fully abstract and compositional.

3 Denotational semantics for CCS

In this section we use our approach from the previous section to endow CCS under (weak) bisimilarity with a fully abstract compositional denotational semantics. We achieve this in two steps:

- first, we consistently modify the CCS transition system to employ traces of actions rather than single actions;
- second, we define the semantic operators on the domain *Proc* from the previous section and show that they reflect the behavior of trace-based CCS.

The first step, although consisting of a purely syntactic transformation, is the actual realization of our insight concerning the semantics of weak bisimilarity, since it emphasizes behavior along arbitrarily long traces. The second step is in many aspects routine, as it merely "institutionalizes" in a final coalgebra the above trace-based behavior. If it were not for the second-order features given by the fixpoint operators, we could have directly employed the general theory from

[18, 24] for this second step – however, fixpoints in combination with arbitrary traces did bring additional difficulties which required ad hoc resolutions, not covered yet, as far as we see, by any general theory of denotational semantics for SOS formats.

Concepts and notations concerning transition systems. We shall consider several *transition system specifications*, which are parameterized by classes *State*, of states, and classes *Label*, of labels and consist of descriptions (of some kind) of labeled transition systems. In our cases:

- *State* will be *Term* or *Proc* and *Label* will be $Act \cup \{\tau\}$, $(Act \cup \{\tau\})^*$ or Act^*;
- the description will consist of either a set of SOS rules (such as CCS and CCST below) or the indication of a coalgebra which yields standardly a labeled transition system (such as $(Proc, unf)$).

Given a transition system specification TS, we write $\vdash_{\mathrm{TS}} s \xrightarrow{l} s'$ to indicate that $s \xrightarrow{l} s'$ is inferable in TS (i.e., belongs to the described labeled transition system). TS has the following associated notions. A relation $\theta \subseteq State \times State$ is called:

- a *strong TS-simulation*, if the following holds for all $(s,t) \in \theta$: if $\vdash_{\mathrm{TS}} s \xrightarrow{l} s'$ for some l and s', then there exists t' such that $\vdash_{\mathrm{TS}} t \xrightarrow{l} t'$ and $(s',t') \in \theta$;
- a *strong TS-bisimulation*, if both θ and $\theta^{\smile}$ are strong TS-simulations.

The *strong TS-bisimilarity* relation, denoted $\sim_{\mathrm{TS}}$, is the union of all strong TS-bisimulations, and is also the largest strong TS-bisimulation.

In case $Label = Act \cup \{\tau\}$, we also have *weak* versions of the above concepts,[3] which we describe below. Given a regular language L over $Act \cup \{\tau\}$, we write $\Vdash_{\mathrm{TS}} s \xrightarrow{L} s'$ to indicate that there exists a word $w = \alpha_1 \ldots \alpha_n \in L$ (with the α_i-s in $Act \cup \{\tau\}$) and $s_0, \ldots, s_n$ such that $s_0 = s$, $s_n = s'$, and $s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ for all $i \in \overline{0, n-1}$. (Notice the usage of the symbol $\Vdash$, distinct from $\vdash$, to denote a notion of "deduction" which is not primitive in TS, but derived from $\vdash$.) Examples of regular languages L include τ^* and $\tau^* \# a \# \tau^*$ for some $a \in Act$. A relation $\theta \subseteq State \times State$ is called:

- a *TS-simulation*, if the following hold for all $(s,t) \in \theta$:
  - if $\Vdash_{\mathrm{TS}} s \xrightarrow{\tau^*} s'$ for some s', then there exists t' such that $\Vdash_{\mathrm{TS}} t \xrightarrow{\tau^*} t'$ and $(s',t') \in \theta$;
  - if $\Vdash_{\mathrm{TS}} s \xrightarrow{\tau^* \# a \# \tau^*} s'$ for some s' and $a \in Act$, then there exists t' such that $\Vdash_{\mathrm{TS}} t \xrightarrow{\tau^* \# a \# \tau^*} t'$ and $(s',t') \in \theta$.
- a *TS-bisimulation*, if both θ and $\theta^{\smile}$ are TS-simulations.

The *TS-bisimilarity* relation, denoted $\approx_{\mathrm{TS}}$, is the union of all TS-bisimulations, and is also the largest TS-bisimulation.

[3] As before, we shall omit the prefix "weak".

The following trace-based characterization of simulation (hence of bisimilarity), as seen in Section 2 crucial for our approach to compositionality, is known already from [14]. Let $del_\tau : (Act \cup \{\tau\})^* \to Act^*$ be the map that deletes all the τ-s. A relation $\theta \subseteq State \times State$ is a TS-simulation iff the following hold for all $(s,t) \in \theta$: if $\vdash_{\mathrm{TS}} s \xrightarrow{v} s'$ for some s' and $v \in (Act \cup \{\tau\})^*$, then there exist t' and $w \in (Act \cup \{\tau\})^*$ such that $del_\tau(v) = del_\tau(w)$, $\vdash_{\mathrm{TS}} t \xrightarrow{w} t'$ and $(s',t') \in \theta$. Thus, according to this characterization, in the bisimilarity game one has to match a trace of actions with a trace "τ-equivalent" to it. Henceforth we take this characterization as the very definition of TS-simulation. (Notice that, in our semantic domain *Proc* from Section 2, τ is absorbed in the monoidal structure, hence "τ-equivalence", that is, "ϵ-equivalence", coincides with *strong* bisimilarity, which in turn coincides, by the internal full abstraction, with equality.)

CCS recalled. For the presentation of CCS, we follow [14] rather closely. We shall not consider the full CCS, but a version restricted in one essential way and in several inessential ways.

The essential restriction consists of allowing only *guarded* sums. For arbitrary sums, (weak) bisimilarity is not a congruence, hence a priori impossible to capture by a compositional denotation. On the other hand, the congruence cogenerated by bisimilarity can receive such a denotation by an adaptation of our approach, but we feel that this would bring technical complications orthogonal to the main ideas of this text. Thus, we take the (rather customary) view that the "real CCS calculus" is that with guarded sums only – already in [14] (at page 113) it is noted: "[weak bisimilarity] is a congruence if we limit the use of Summation to *guarded* sums [...]. In fact, in applications we hardly ever use Summation in any other way." (Also, unguarded sums are excluded already at the syntax level in monographs (of related calculi) such as [20].)

The inessential restrictions consist of excluding the *restriction* and *renaming* operators and dealing with simple (non-mutual) recursion only. These features are of course essential to the calculus, but inessential to the presentation of our semantics, since it is immediate how they can be handled in our framework.

On the other hand, we do not require the fixpoint expressions to be guarded. Guardedness is a convenient/desired feature in the context of strong bisimilarity, since there it guarantees (in the presence of finite summations) finite branching. But since weak bisimilarity is infinitely branching anyway, here unguarded replication does not raise new fundamental problems, and yields perfectly valid behavior – for example, the π-calculus-style replication $!P$ is only definable by an unguarded fixpoint expression, $\mu X.\, X|P$.

Henceforth, we call "CCS" the indicated restricted version of CCS, presented below in detail. We fix the following:

- A set *Act*, of *loud actions*, ranged over by a; an item $\tau \notin Act$, called the *silent action*; $Act \cup \{\tau\}$ is called the set of *actions* and is ranged over by α;
- A map $^- : Act \to Act$ such that $\overline{\overline{a}} = a$ for all $a \in Act$.
- An infinite set *Var*, of (process) variables, ranged over by X, Y.

– A set *Index*, of *indexes* (for summations), ranged over by i, j; I will range over $\mathcal{P}(Index)$. (*Index* is not required to be finite.)

The set *Term*, of *(process) terms*, ranged over by P, Q, is given by the following grammar:

$$P \ ::= \ X \mid \sum_{i \in I} \alpha_i . P_i \mid P|Q \mid \mu X . P$$

In a term $\mu X . P$, X is bound in P – this notion of binding yields standardly a notion of alpha-equivalence on terms. We identify terms modulo alpha-equivalence. $P[Q/X]$ denotes the term obtained from P by (capture-free) substituting all free variables of P with Q. $P[Q/X]^n$ denotes $P[Q/X] \ldots [Q/X]$ (n times).

The CCS transition system infers triples $P \xrightarrow{\alpha} P'$:

$$\frac{\cdot}{\sum_{i \in I} \alpha_i . P_i \xrightarrow{\alpha_j} P_j} \ [j \in I] \ (\mathsf{Sum}) \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} (\mathsf{ParL}) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'} (\mathsf{ParR})$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} (\mathsf{Com}) \qquad \frac{P[(\mu X . P)/X] \xrightarrow{\alpha} P'}{\mu X . P \xrightarrow{\alpha} P'} (\mathsf{Rec})$$

Trace-based CCS. We first employ our insight concerning arbitrarily long traces for compositional bisimilarity *at the syntactic level*, by defining the following system CCST, a variation of CCS with *traces* instead of single actions as labels. (In the rule (ParT) below, $| : Act^* \times Act^* \to \mathcal{P}(Act^*)$ is the synchronized shuffle defined in Section 2.)

$$\frac{\cdot}{P \xrightarrow{\epsilon} P} (\mathsf{Silent}) \qquad \frac{P_j \xrightarrow{w} P'}{\sum_{i \in I} \alpha_i . P_i \xrightarrow{\alpha_j \# w} P'} \ [j \in I] \ (\mathsf{SumT})$$

$$\frac{P \xrightarrow{w_1} P' \quad Q \xrightarrow{w_2} Q'}{P|Q \xrightarrow{w} P'|Q'} \ [w \in w_1|w_2] \ (\mathsf{ParT}) \qquad \frac{P[(\mu X . P)/X] \xrightarrow{w} P'}{\mu X . P \xrightarrow{w} P'} (\mathsf{RecT})$$

The rules of CCST were produced by taking the reflexive-transitive closure of the rules of CCS, i.e., by composing the system CCS with itself *horizontally* an indefinite number of times. In the process, we also made sure that zero or more τ actions were identified with the empty trace ϵ, and in particular we have added the rule (Silent).

Lemma 1. *$\vdash_{\mathrm{CCS}}$ is closed under the rules that define $\vdash_{\mathrm{CCST}}$ (modulo the absorbtion of τ for the case of parallel composition), namely:*
(1) $\vdash_{\mathrm{CCS}} P \xrightarrow{\epsilon} P$.
(2) If $j \in I$ *and* $\vdash_{\mathrm{CCS}} P_j \xrightarrow{w} P'$, *then* $\vdash_{\mathrm{CCS}} \sum_{i \in I} \alpha_i . P_i \xrightarrow{\alpha_j \# w} P'$.
(3) If $\vdash_{\mathrm{CCS}} P \xrightarrow{w_1} P'$, $\vdash_{\mathrm{CCS}} Q \xrightarrow{w_2} Q'$, *and* $v \in del_\tau(w_1)|del_\tau(w_2)$, *then there exists* w *such that* $del_\tau(w) = v$ *and* $\vdash_{\mathrm{CCS}} P|Q \xrightarrow{w} P'|Q'$.
(4) If $\vdash_{\mathrm{CCS}} P[(\mu X . P)/X] \xrightarrow{w} P'$, *then* $\vdash_{\mathrm{CCS}} \mu X . P \xrightarrow{w} P'$.

Proof. (1): Immediate. (2) and (4): By case analysis on whether w is empty.
(3): By an easy (but tedious) induction on the sum of the lengths of w_1 and w_2, using the definition of $|$ on traces. □

The next proposition states the fact we naturally expect from CCST – that it indeed produces precisely the traces of actions produced by CCS, *with the τ-actions absorbed*.

Proposition 1. *The following hold for all $P, P' \in Term$, $w \in (Act \cup \{\tau\})^*$, and $v \in Act^*$:*
(1) $\vdash_{\mathrm{CCS}} P \xrightarrow{w} P'$ *implies* $\vdash_{\mathrm{CCST}} P \xrightarrow{del_\tau(w)} P'$.
(2) $\vdash_{\mathrm{CCST}} P \xrightarrow{v} P'$, *then there exists* $w' \in (Act \cup \{\tau\})^*$ *such that* $del_\tau(w') = v$ *and* $\vdash_{\mathrm{CCS}} P \xrightarrow{w'} P'$.

Proof. (1): Easy induction on w. (2): Immediate consequence of Lemma 1. □

As a consequence, we have the identity between CCS-bisimilarity and strong CCST-bisimilarity:

Corollary 1. *For all $P, Q \in Term$, $P \approx_{\mathrm{CCS}} Q$ iff $P \sim_{\mathrm{CCST}} Q$.*

Proof. Immediate from Proposition 1 (using the aforementioned trace-based characterization of bisimilarity). □

Semantic domain and operators. The semantic domain is the one advocated in Section 2, namely the carrier set *Proc* of the final monoidal multi-step coalgebra $(Proc, unf : Proc \to \mathcal{P}(Proc \times Act^*))$.

As for the semantic operators, the ones corresponding to the summation and parallel composition syntactic constructs have *almost* necessary definitions induced automatically by the de Simone SOS format of the CCST rules that involve these constructs:
- For each $I \subseteq Index$, $\mathsf{Sum}_I : (I \to (Act \cup \{\epsilon\}) \times Proc) \to Proc$, by
$unf(\mathsf{Sum}_I((\alpha_i, \pi_i)_{i \in I})) = \{(\epsilon, \mathsf{Sum}_I((\alpha_i, \pi_i)_{i \in I}))\} \cup \{(\alpha_i \# w, \pi'). i \in I \wedge (w, \pi') \in unf(\pi_i)\}$;
- $\mathsf{Par} : Proc \times Proc \to Proc$, by
$unf(\mathsf{Par}(\pi_1, \pi_2)) = \{(w, \mathsf{Par}(\pi_1', \pi_2')). \exists w_1, w_2. (w_1, \pi_1) \in unf(\pi_1') \wedge (w_2, \pi_2) \in unf(\pi_2') \wedge w \in w_1 | w_2\}$ (where again $| : Act^* \times Act^* \to \mathcal{P}(Act^*)$ is the synchronized shuffle defined in Section 2).

Notice that, in the case of summation, we have explicitly added a self-ϵ-transition to ensure reflexivity of the result. The other properties, namely, transitivity and prefix-closeness, can be easily seen to be preserved by Sum_I. Moreover, Par is easily seen to preserve monoidality. These mean that the definitions of these operators are *correct corecursive definitions on Proc*.

In order to define a semantic operator corresponding to the μ construct, we cannot simply transliterate the corresponding rule (RecT) as we did for the other operators. Indeed, (RecT) is not in de Simone or in other amenable format that would allow a local semantic definition. However, (RecT) can be replaced with the following:

$$\frac{P[P/X]^n \xrightarrow{w} Q'}{\mu X.\, P \xrightarrow{w} Q'[(\mu X.\, P)/X]}(\mathsf{RecT'})$$

Intuitively, the recursive unfolding of a (sub)term $\mu X.\, P$ in a derivation needs to be performed only once, atomically, for a sufficiently large depth n.

In the next lemma (and henceforth) we write $\vdash_{\mathrm{CCST},k} P \xrightarrow{w} Q$ to indicate the fact that $P \xrightarrow{w} Q$ has a derivation of height *at most* k in CCST.

Lemma 2. *(1) If* $\vdash_{\mathrm{CCST},k} P[(\mu X.\, P)/X] \xrightarrow{w} P'$*, then there exist* $n \in \mathbb{N}$ *and* Q' *such that* $Q'[(\mu X.\, P)/X] = P'$ *and* $\vdash_{\mathrm{CCST},k} P[P/X]^n \xrightarrow{w} Q'$*.*
(2) If $\vdash_{\mathrm{CCST}} P[P/X]^n \xrightarrow{w} P'$*, then* $\vdash_{\mathrm{CCST}} \mu X.\, P \xrightarrow{w} P'[(\mu X.\, P)/X]$

Proof. (1): By induction on k. (2) By induction on the definition of $\vdash_{\mathrm{CCST}}$. □

Now we are ready to define the semantic operator $\mathsf{Fix} : (\mathit{Proc} \to \mathit{Proc}) \to \mathit{Proc}$ (corresponding to μ) corecursively:
$\mathit{unf}(\mathsf{Fix}(F)) = \bigcup_{n \geq 1} \{(w, F'(\mathsf{Fix}(F))).\, \forall \pi \in \mathit{Proc}.\, (w, F'\, \pi) \in \mathit{unf}(F^n\, \pi)\}$.
(where $F : \mathit{Proc} \to \mathit{Proc}$ and F^n denotes $F \circ \ldots \circ F$ (n times))

Again, one can immediately check that the above is a correct definition. The definition is a rather standard semantic representation of the rule (RecT'), with the opening of the scope of X (regarding (RecT') as being applied backwards) handled semantically via the interplay between functional binding and universal quantification. On the other hand, the reader may legitimately wonder whether a more abstract semantic definition of recursion, namely one employing genuine fixpoints instead of numbers, would be possible. Unfortunately, an operator on processes whose (least) fixed point (according to some ordering) would give our semantic recursive item does not seem to be available, because of the non-continuous behavior of infinitely-branching recursion, which expands the trees *locally*. By contrast, a domain-theoretic setting (not suitable here due to infinite branching) allows itself to take *global* approximating snapshots from the recursively growing trees, enabling the mentioned more elegant treatment of recursion. (In the context of infinite branching, such snapshots would not be enough to characterize the final trees.)

The coalgebra $(\mathit{Proc}, \mathit{unf})$ yields standardly an Act^*-labeled transition system with processes as states: $\vdash_{\mathit{Proc}} \pi \xrightarrow{w} \pi'$ iff $(w, \pi') \in \mathit{unf}(\pi)$. It is convenient (including more readable) to rephrase the above coinductive definitions in terms of this transition system:

- (C1) $\vdash_{\mathit{Proc}} \mathsf{Sum}_I((\alpha_i, \pi_i)_{i \in I}) \xrightarrow{w} \pi'$ iff $(w = \epsilon \ \wedge \ \pi' = \mathsf{Sum}_I((\alpha_i, \pi_i)_{i \in I})) \vee (\exists i \in I, w'.\, w = \alpha_i \# w' \wedge \vdash_{\mathit{Proc}} \pi_i \xrightarrow{w'} \pi')$.
- (C2) $\vdash_{\mathit{Proc}} \mathsf{Par}(\pi_1, \pi_2) \xrightarrow{w} \pi'$ iff $\exists w_1, w_2, \pi_1', \pi_2'.\, w \in w_1 | w_2 \wedge \pi' = \mathsf{Par}(\pi_1', \pi_2') \wedge \vdash_{\mathit{Proc}} \pi_1 \xrightarrow{w_1} \pi_1' \wedge \vdash_{\mathit{Proc}} \pi_2 \xrightarrow{w_2} \pi_2'$.
- (C3) $\vdash_{\mathit{Proc}} \mathsf{Fix}(F) \xrightarrow{w} \pi'$ iff $\exists F', n \geq 1.\, \pi' = F'(\mathsf{Fix}(F)) \wedge (\forall \pi.\, F^n\, \pi \xrightarrow{w} F'\, \pi)$.

Everything is now almost set in place for the proof of full abstraction, except for one final detail: in (RecT'), we need to make sure that $P[P/X]^n$ is smaller than $\mu X.\, P$ in some way, so that $\mu X.\, P$ is simplified by an application of (RecT'). To this end, we define the relation $\succ \subseteq \mathit{Term} \times \mathit{Term}$ as follows: $P \succ Q$ iff one of the following holds:

- $\exists Q'.\, P = Q|Q' \ \vee \ P = Q'|Q$;
- $\exists I, (\alpha_i)_{i\in I}, (P_i)_{i\in I}, j \in I.\, P = \sum_{i\in I} \alpha_i.\, P_i \ \wedge \ Q = P_j$;
- $\exists R, X, n.\, P = \mu X.\, R \ \wedge \ Q = R[R/X]^n$.

Lemma 3. *$\succ$ is well-founded, in that there exists no infinite sequence $(P_k)_{k\in\mathbb{N}}$ such that $\forall k.\, P_k \succ P_{k+1}$.*

Proof. Define $\rightsquigarrow \subseteq \mathit{Term} \times \mathit{Term}$ inductively by the following clauses:
- $\mu X.\, P \ \rightsquigarrow \ P[P/X]^n$;
- If $P \ \rightsquigarrow \ P'$, then $P|Q \ \rightsquigarrow \ P'|Q$ and $Q|P \ \rightsquigarrow \ Q|P'$;
- If $j \in I$ and $P \ \rightsquigarrow \ P'$, then $\sum_{i\in I} P_i \ \rightsquigarrow \ \sum_{i\in I} P'_i$, where $P'_j = P'$ and $P'_i = P_i$ form all $i \neq j$.

Let $WF_{\rightsquigarrow}$ be the set of all terms well-founded w.r.t. $\rightsquigarrow$, i.e., of all terms P for which there exists no infinite sequence $(P_k)_{k\in\mathbb{N}}$ such that $P_0 = P$ and $\forall k.\, P_k \ \rightsquigarrow \ P_{k+1}$.

Claim1: If $P \in WF_{\rightsquigarrow}$ and $\forall X.\, \sigma X \in WF_{\rightsquigarrow}$, then $P[\sigma] \in WF_{\rightsquigarrow}$ (where σ is any map in $\mathit{Var} \to \mathit{Term}$ and $P[\sigma]$ is the term obtained from P by applying the substitution σ). (Proof: By induction on P.)

Claim2: If $P \in WF_{\rightsquigarrow}$, then $P[P/X]^n \in WF_{\rightsquigarrow}$. (Proof: By induction on n, using Claim1.)

Claim3: $\rightsquigarrow$ is well-founded. (Proof: By induction on terms, using Claim2 for μ-terms.)

Now, we can prove that $\succ$ is well-founded: Assume an infinite sequence $(P_k)_{k\in\mathbb{N}}$ such that $\forall k.\, P_k \succ P_{k+1}$. We construct, recursively on k, a sequence $(Q_k)_{k\in\mathbb{N}}$ such that $\forall k.\, Q_k \ \rightsquigarrow \ Q_{k+1}$, thus deriving a contradiction with Claim3. In fact, we define the Q_k's via the contexts (i.e., terms with one hole) $C_k[*]$, by $Q_k = C_k[P]$, where the $C_k[*]$-s are defined as follows:
- $C_0[*] = *$ (the trivial context);
- if $P_{n+1} = R|P_n$, then $C_{n+1}[*] = R \,|\, C_n[*]$;
- if $P_{n+1} = P_n|R$, then $C_{n+1}[*] = C_n[*] \,|\, R$;
- if $P_{n+1} = \sum_{i\in I} \alpha_i.\, R_i$, $j \in I$ and $P_n = R_j$, then $C_{n+1}[*] = \sum_{i\in I} \alpha_i.\, H_i$, where $H_j = C_n[*]$ and $H_j = R_j$ for all $j \neq i$;
- if $P_{n+1} = \mu X.\, R$ and $P_n = R[R/X]^n$, then $C_{n+1}[*] = C_n[*]$.

It is easy to see that indeed the Q_k-s (i.e., the $C_k[P_k]$-s) form a $\rightsquigarrow$-chain. □

Denotational semantics. The above semantic operators yield standardly an interpretation $\langle \_ \rangle : \mathit{Term} \to (\mathit{Var} \to \mathit{Proc}) \to \mathit{Proc}$ of the CCS syntax in environments (via a higher-order version of the initial algebra semantics):

- $\langle \sum_{i\in I} \alpha_i.P_i \rangle\, \rho = \mathsf{Sum}_I((\alpha_i, \langle P_i \rangle\, \rho)_{i\in I})$;
- $\langle P|Q \rangle\, \rho = \mathsf{Par}(\langle P \rangle\, \rho, \langle Q \rangle\, \rho)$;
- $\langle \mu X.\, P \rangle\, \rho = \mathsf{Fix}(\lambda\pi.\, \langle P \rangle(\rho[X \leftarrow \pi]))$.

Lemma 4. *The following hold for all $X \in \mathit{Var}$, $P, Q \in \mathit{Term}$, $\pi \in \mathit{Proc}$, $\rho, \rho' : \mathit{Var} \to \mathit{Proc}$ and $n \in \mathbb{N}$:*
(1) If $\rho\, Y = \rho'\, Y$ for all $Y \in FV(P)$, then $\langle P \rangle\, \rho = \langle P \rangle\, \rho'$.
(2) $\langle P[Q/X] \rangle\, \rho = \langle P \rangle(\rho[X \leftarrow \langle Q \rangle\, \rho])$.
(3) $\langle P[P/X]^n \rangle(\rho[X \leftarrow \pi]) = F^{n+1}\, \pi$, where $F = \lambda\pi.\, \langle P \rangle(\rho[X \leftarrow \pi])$.

Proof. Points (1) and (2) state well-known facts holding for any standard interpretation of syntax with static bindings in a domain (as is the case here). We prove point (3) by induction on n. The base case, $n = 0$, follows from point (2). For the inductive case, we have the following chain of equalities:
$\langle P[P/X]^{n+1}\rangle(\rho[X \leftarrow \pi]) = \langle P[P/X]^n[P/X]\rangle(\rho[X \leftarrow \pi]) =$ (by point (2)) $=$
$\langle P[P/X]^n\rangle(\rho[X \leftarrow \pi][X \leftarrow \langle P\rangle(\rho[X \leftarrow \pi])]) = \langle P[P/X]^n\rangle(\rho[X \leftarrow \langle P\rangle(\rho[X \leftarrow \pi])]) =$ (by IH) $= F^{n+1}(\langle P\rangle(\rho[X \leftarrow \pi])) = F^{n+1}(F\,\pi) = F^{n+2}\,\pi$. □

We are finally ready to state the connection between CCST and *Proc* (which leads to full abstraction):

Proposition 2. *The following hold for all $P, P' \in Term$, $\pi' \in Proc$, $w \in Act^*$, and $\rho : Var \to Proc$:*
(1) If $\vdash_{\mathrm{CCST}} P \xrightarrow{w} P'$, then $\vdash_{Proc} \langle P\rangle\,\rho \xrightarrow{w} \langle P'\rangle\,\rho$.
(2) If $\vdash_{Proc} \langle P\rangle\,\rho \xrightarrow{w} \pi'$, then there exists P'' such that $\vdash_{\mathrm{CCST}} P \xrightarrow{w} P''$ and $\pi' = \langle P''\rangle\,\rho$.
(In other words, for every ρ, the map $P \mapsto \langle P\rangle\,\rho$ is a morphism between the multi-step coalgebra induced by CCST and $(Proc, unf)$.)

Proof. (1): We prove by induction on k that $\vdash_{\mathrm{CCST},k} P \xrightarrow{w} P'$ implies $\forall\rho.\ \vdash_{Proc} \langle P\rangle\,\rho \xrightarrow{w} \langle P'\rangle\,\rho$. The cases where the last applied rule was (SumT) or (ParT) are immediate. We only treat the case of (RecT).

Assume $\vdash_{\mathrm{CCST},k} P[(\mu X.\,P)/X] \xrightarrow{w} P'$. Fix ρ. We need to show
$\vdash_{Proc} \langle \mu X.\,P\rangle\,\rho \xrightarrow{w} \langle P'\rangle\,\rho$. Let $F = \lambda\pi.\,\langle P\rangle(\rho[X \leftarrow \pi])$.

Fix π. By Lemma 2.(1), there exist n and Q' such that
$P' = Q'[(\mu X.\,P)/X]$ and $\vdash_{\mathrm{CCST},k} P[P/X]^n \xrightarrow{w} Q'$.
Let $F' = \lambda\pi.\,\langle Q'\rangle(\rho[X \leftarrow \pi])$. With IH, we have
$\vdash_{Proc} \langle P[P/X]^n\rangle(\rho[X \leftarrow \pi]) \xrightarrow{w} \langle Q'\rangle(\rho[X \leftarrow \pi])$, i.e.,
$\vdash_{Proc} \langle P[P/X]^n\rangle(\rho[X \leftarrow \pi]) \xrightarrow{w} F'\,\pi$,
hence, using Lemma 4.(3), $\vdash_{Proc} F^{n+1}\pi \xrightarrow{w} F'\,\pi$.

We thus proved $\forall\pi.\ \vdash_{Proc} F^{n+1}\pi \xrightarrow{w} F'\,\pi$, hence, with (C3),
$\vdash_{Proc} \mathsf{Fix}(F) \xrightarrow{w} F'(\mathsf{Fix}(F))$. Moreover, by Lemma 4.(2), we have $F'(\mathsf{Fix}(F)) =$
$\langle Q'\rangle(\rho[X \leftarrow \mathsf{Fix}(F)]) = \langle Q'\rangle(\rho[X \leftarrow \langle \mu X.\,P\rangle\,\rho]) = \langle Q'[(\mu X.P)/X]\rangle(\rho) = \langle P'\rangle\,\rho$.
Ultimately, we obtain $\vdash_{Proc} \langle \mu X.\,P\rangle\,\rho \xrightarrow{w} \langle P'\rangle\,\rho$, as desired.
(2): By well-founded induction on P (the lefthand side of the hypothesis) w.r.t. $\succ$ (since $\succ$ is well-founded according to Lemma 3). Again, the cases where P is a summation or a parallel composition are easy. We only consider the case of recursion.

Assume $\langle \mu X.\,P\rangle\,\rho \xrightarrow{w} \pi'$. Let $F = \lambda\pi.\,\langle P\rangle(\rho[X \leftarrow \pi])$. Then $\langle \mu X.\,P\rangle\,\rho = \mathsf{Fix}(F)$, hence Fact0: $\mathsf{Fix}(F) \xrightarrow{w} \pi'$. By Lemma 4.(3), we have
Fact1: $\langle P[P/X]^n\rangle\,(\rho[X \leftarrow \mathsf{Fix}(F)]) = F^n(\mathsf{Fix}(F))$,

Moreover, from Fact0 and (C3), we obtain $n \geq 1$ and F' such that
$\pi' = F'(\mathsf{Fix}(F))$ and $\forall\pi.\ \vdash_{Proc} F^n\,\pi \xrightarrow{w} F'\,\pi$. In particular,
$\vdash_{Proc} F^n(\mathsf{Fix}(F)) \xrightarrow{w} F'(\mathsf{Fix}(F))$, i.e., $\vdash_{Proc} F^n(\mathsf{Fix}(F)) \xrightarrow{w} \pi'$, hence, with Fact1,
$\vdash_{Proc} \langle P[P/X]^n\rangle\,(\rho[X \leftarrow \mathsf{Fix}(F)]) \xrightarrow{w} \pi'$.

With IH, we find P' such that $\langle P' \rangle\,(\rho[X \leftarrow \mathsf{Fix}(F)]) = \pi'$ and $\vdash_{\mathrm{CCST}} P[P/X]^n \xrightarrow{w} P'$. By Lemma 2.(2), we have $\vdash_{\mathrm{CCST}} \mu X.\, P \xrightarrow{w} P'[(\mu X.\, P)/X]$.

Moreover, by Lemma 4.(2), we have $\langle P'[(\mu X.\, P)/X] \rangle\, \rho = \langle P' \rangle\,(\rho[X \leftarrow \langle \mu X.\, P \rangle\, \rho])$ $= F'(\mathsf{Fix}(F)) = \pi'$, making $P'[(\mu X.\, P)/X]$ the desired term. □

Corollary 2. *The following are equivalent for all $P, Q \in Term$: $P \sim_{\mathrm{CCST}} Q$ iff $\langle P \rangle = \langle Q \rangle$.*

Proof. Immediately from Proposition 2 (since coalgebra morphisms preserve and reflect strong bisimilarity, and since strong bisimilarity in *Proc* is equality). □

Now, Corollaries 1 and 2 immediately yield full abstraction:

Theorem 1. *The following are equivalent for all $P, Q \in Term$: $P \approx_{\mathrm{CCS}} Q$ iff $\langle P \rangle = \langle Q \rangle$.*

4 Conclusions and related work

We described an approach to (weak) bisimilarity and illustrated it by providing a novel denotational semantics for CCS under bisimilarity which is both *fully abstract* and *compositional*. Previous approaches to the denotational semantics of bisimilarity CCS [2], as well as instances of more generic approaches [18] that can in principle cover this case, essentially consider final *one-step* coalgebras for the semantic domain. Other works on the denotation of related calculi either do not feature compositionality [12, 13] or cover only strong bisimilarity and related strong equivalencies [10, 22, 4, 23].

Models for process calculi which are *intensional* (in that the identity of the semantic items does not coincide with bisimilarity or other targeted notion of operational equivalence) were proposed in [8, 16, 9, 7, 21] (among other places). The framework of [9] (extending the one of [16]) offers facilities to define and reason "syntax freely" about weak bisimilarity in models which are already fully abstract w.r.t. strong bisimilarity, via a suitable hiding functor. The relationship between the abstract categorical machinery from there aimed at hiding the τ-actions and our packing and unpacking bijections between processes and their compact representations deserves future research.

Our technique of combining traces with coalgebra seems able to capture bisimilarity of a wide range of process calculi. In particular, our transformation of CCS into CCST could be soundly performed on systems in an SOS format of a quite general kind, e.g., of the kind singled out in [5, 25] to ensure that weak bisimilarity is a congruence. We have not worked out the details yet though.

With a bit of extra effort, we could have finetuned our definitions into domain-theoretic ones using Abramsky powerdomains instead of powersets, along the lines of [10, 22]. However, our semantics would then have not captured bisimilarity, but the weaker relation of having all finite subtrees bisimilar [1].

References

1. S. Abramsky. A domain equation for bisimulation. *Inf. Comput.*, 92(2):161–218, 1991.
2. P. Aczel. Final universes of processes. In *MFPS'93*, pages 1–28, 1993.
3. M. Baldamus, J. Parrow, and B. Victor. A fully abstract encoding of the *pi*-calculus with data terms. In *ICALP'05*, pages 1202–1213, 2005.
4. M. Baldamus and T. Stauner. Modifying Esterel concepts to model hybrid systems. *Electr. Notes Theor. Comput. Sci.*, 65(5), 2002.
5. B. Bloom. Structural operational semantics for weak bisimulations. *Theor. Comput. Sci.*, 146(1&2):25–68, 1995.
6. M. Boreale and F. Gadducci. Denotational testing semantics in coinductive form. In B. Rovan and P. Vojtás, editors, *MFCS*, volume 2747 of *Lecture Notes in Computer Science*, pages 279–289, 2003.
7. M. G. Buscemi and U. Montanari. A first order coalgebraic model of pi-calculus early observational equivalence. In *CONCUR*, pages 449–465, 2002.
8. G. L. Cattani and P. Sewell. Models for name-passing processes: interleaving and causal. In *LICS 2000*, pages 322–333, 2000.
9. M. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps. In *LICS'99*, pages 67–76, 1999.
10. M. P. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *LICS'96*, pages 43–54, 1996.
11. M. Hennessy. A fully abstract denotational semantics for the π-calculus. *Theor. Comput. Sci.*, 278(1-2):53–89, 2002.
12. F. Honsell, M. Lenisa, U. Montanari, and M. Pistore. Final semantics for the pi-calculus. In *PROCOMET'98*, pages 225–243, 1998.
13. M. Lenisa. *Themes in Final Semantics.* Dipartimento di Informatica, Universita' di Pisa, TD 6, 1998.
14. R. Milner. *Communication and concurrency.* Prentice Hall, 1989.
15. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts i and ii. *Inf. Comput.*, 100(1):1–77, 1992.
16. M. Nielsen and A. Chang. Observe behaviour categorically. In *FST&TCS'95*, pages 263–278, 1995.
17. A. Popescu. A fully abstract coalgebraic semantics for the pi-calculus under weak bisimilarity. Tech. Report UIUCDCS-R-2009-3045. University of Illinois, 2009.
18. J. J. M. M. Rutten. Processes as terms: Non-well-founded models for bisimulation. *Math. Struct. Comp. Sci.*, 2(3):257–275, 1992.
19. J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.
20. D. Sangiorgi and D. Walker. *The π-calculus. A theory of mobile processes.* Cambridge, 2001.
21. A. Sokolova, E. P. de Vink, and H. Woracek. Weak bisimulation for action-type coalgebras. *Electr. Notes Theor. Comput. Sci.*, 122:211–228, 2005.
22. I. Stark. A fully-abstract domain model for the π-calculus. In *LICS'96*, pages 36–42, 1996.
23. S. Staton. *Name-passing process calculi: operational models and structural operational semantics.* PhD thesis, University of Cambridge, 2007.
24. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *LICS'97*, pages 280–291, 1997.
25. R. J. van Glabbeek. On cool congruence formats for weak bisimulations. In D. V. Hung and M. Wirsing, editors, *ICTAC*, volume 3722 of *Lecture Notes in Computer Science*, pages 318–333, 2005.