Extensions of standard weak bisimulation machinery: finite-state general processes, refinable actions, maximal-progress and time [1]

Mario Bravetti

Università di Bologna, Dipartimento di Scienze dell'Informazione, Mura Anteo Zamboni 7, 40127 Bologna, Italy, E-mail: bravetti@cs.unibo.it

Abstract

We present our work on extending the standard machinery for weak bisimulation to deal with: finite-state processes of calculi with a full signature, including static operators like parallel; semantic action refinement and ST bisimulation; maximal-progress, i.e. priority of standard actions over unprioritized actions; representation of time: discrete real-time and Markovian stochastic time. For every such topic we show that it is possible to resort simply to weak bisimulation and that we can exploit this to obtain, via modifications to the standard machinery: finite-stateness of semantic models when static operators are not replicable by recursion, as for CCS with the standard semantics, thus yielding decidability of equivalence; structural operational semantics for terms; a complete axiomatization for finite-state processes via a modification of the standard theory of standard equation sets and of the normal-form derivation procedure.

1 Introduction

This paper presents work that we have done on how to extend standard machinery to cope with several extensions of standard process algebra. Every extension is presented by using the same structure: basic ideas on how to resort to standard machinery and definition of operational semantics; set of axioms that compose the axiomatization; technique to prove completeness: variation to the standard theory of standard equation sets and to the procedure for deriving normal forms from terms.

The standard machinery for basic CCS with recursion [15], i.e. for a calculus with prefix $\alpha.P$ (where α is either a visible action a or τ), choice $P+Q$ and recursion $recX.P$, and full CCS without recursion [14], i.e. when also static

[1] Research partially funded by EU Integrated Project Sensoria, contract n. 016004.

operators like parallel $P|Q$ and restriction $P\backslash L$ are considered, is presented at the beginning of the paper (in Sect. 2).

We then show (in Sect. 3) how to extend such machinery to finite-state processes of a full calculus that includes *both static operators and recursion*, more precisely to processeses such that static operators are not replicable by recursion [2,3,1]. The basic idea is to *express hiding mechanisms explicitely* via an hiding operator P/L (like e.g. the hiding mechanism that is part of the CCS parallel operator) and to introduce a *new axiom for exchange of hiding and recursion* and a new normal-form derivation procedure that combines unguardedness removal with static operator elimination in an inductive way. The work presented in Sect. 3 has been done in cooperation with Jos Beaten.

The approach is further extended (in Sect. 4) to deal with actions that can be refined by means of a *semantic action refinement* operator "$P[a \rightsquigarrow Q]$", where every visible action "a" executed by P is refined by a term "Q" [6,5]. The basic idea is to express *weak ST bisimulation* (needed to get a congruence in the presence of refinement) via standard weak bisimulation by using: the *dynamic name technique* or *the stack technique* to obtain semi-actions with indexes in semantic models in a way that preserves finite-stateness and the *levelwise reindexing technique* to be able to express the two semantics above in a compositional way (in Structural Operational Semantics and in axioms for parallel).

We then (in Sect. 5) endow basic CCS (with recursion) with an unprioritized action σ to express *maximal progress* [7,5,8]. The basic idea is to introduce an auxiliary operator $pri(P)$ that denotes the *scope of priority*. Such an operator is used to produce a new version for the axioms that allow weak unguardedness to be removed: a new unguardedness removal procedure is produced. The work presented in Sect. 4 and 5 has been done in cooperation with Roberto Gorrieri.

Finally, (in Sect. 6) we extend the basic calculus with maximal progress above with static operators by interpreting σ actions to be timed actions [4,8]. More precisely, we consider two different kinds of time: discrete real-time, where σ is a "tick" and Markovian stochastic time, where σ is a positive real number representing a rate of an exponential distribution. The basic ideas are: make use of a parallel operator that (for both kinds of time) is compatible with standard weak bisimulation, i.e. is such that a process can execute a σ action only when other processes in parallel with it *can explicitly execute a σ action* too; show that in this context we can use (when merging standard equation sets) an equivalence implied by standard weak bisimulation that can be expressed in a *much simpler way* by considering just strong "σ" moves instead of weak "σ" moves (by exploiting time determinism in discrete real time and stochastic delay summation in Markovian stochastic time); and perform *τ saturation after σ prefixes* in normal forms before showing that they are equated by the axiomatization via standard equation sets.

The paper is concluded (in Sect. 7) by reporting possible applications and extensions of the machinery produced.

2 Standard Machinery

The set of action names $\mathcal{A}$ is ranged over by $a, b, c, \ldots$. The set of actions $Act = \mathcal{A} \cup \{\overline{a} \mid a \in \mathcal{A}\} \cup \{\tau\}$, which includes co-actions and the silent action τ denoting an internal move, is ranged over by $\alpha, \alpha', \ldots$. The set of term variables is Var, ranged over by $X, Y, \ldots$. The set $\mathcal{E}$ of CCS behavior expressions, ranged over by E, F is defined by the following syntax.

$$E ::= \underline{0} \mid X \mid \alpha.E \mid E + E \mid E|E \mid E\backslash L \mid E[\varphi] \mid recX.E$$

The meaning of the operators is the standard one of [14,15], where "$recX.E$" denotes recursion. Closed terms are terms that do not include free variables (i.e. variables X not bound by a "$recX.E$" operator) and are called *processes*, ranged over by P,Q.

A a variable X is *serial* in E if every free occurrence of X in E (if any) is in the scope of operators $\alpha.F$, $F' + F''$ or $recY.F$ (for any variable Y) only. Furthermore, a variable X that is serial in E is also: *weakly guarded* if every free occurrence of X in E is in the scope of an operator $\alpha.F$, *strongly guarded* (or simply *guarded*) if it additionally holds that $\alpha \neq \tau$, *fully unguarded* if it is not *weakly guarded*, *fully unguarded* if it is not *strongly guarded*.

Basic CCS is the calculus considered in [15], i.e. the set of terms of CCS obtained by considering a restricted syntax where *static* operators (i.e. "$E|E$", "$E\backslash L$" and "$E[\varphi]$") are not included. In particular here we will distinguish between basic CCS without recursive definitions (i.e. excluding from the CCS syntax also the "$recX.E$" operator and variables X), simply called basic CCS and denoted by $BCCS$, and basic CCS with recursive definitions, denoted by $BCCS_{rec}$. A recursion "$recX.F$" occurring in a $BCCS_{rec}$ term E is a *guarded recursion* if X is guarded in F. We use $BCCS_{grec}$ to denote the subset of $BCCS_{rec}$ terms that include guarded recursive definitions only.

The standard structural operational semantics for $BCCS_{rec}$ operators and CCS static operators is presented in Tables 1 and 2, respectively. $type(\alpha)$ yields the name in $\mathcal{A}$ of the action α or τ if α is the silent action.

As in [15] we use $\xrightarrow{\alpha}^*$ to denote computations composed of all $\xrightarrow{\alpha}$ transitions whose length is possibly zero. Let $\overset{\alpha}{\Longrightarrow}$ denote $\xrightarrow{\tau}^* \xrightarrow{\alpha} \xrightarrow{\tau}^*$. Moreover we define $\overset{\hat{\alpha}}{\Longrightarrow} = \overset{\alpha}{\Longrightarrow}$ if $\alpha \neq \tau$ and $\overset{\hat{\tau}}{\Longrightarrow} = \xrightarrow{\tau}^*$.

Definition 2.1 A relation β over processes is a weak bisimulation if, whenever $(P, Q) \in \beta$:

- If $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \overset{\hat{\alpha}}{\Longrightarrow} Q'$ and $(P', Q') \in \beta$.
- If $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \overset{\hat{\alpha}}{\Longrightarrow} P'$ and $(P', Q') \in \beta$.

Two processes P, Q are weakly bisimilar, written $P \approx Q$, iff (P, Q) is included in some weak bisimulation. ■

$$\alpha.P \xrightarrow{\alpha} P$$

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\frac{P\{recX.P/X\} \xrightarrow{\alpha} P'}{recX.P \xrightarrow{\alpha} P'}$$

Table 1
Structural operational rules for $BCCS_{rec}$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L}\ type(\alpha) \notin L \qquad \frac{P \xrightarrow{\alpha} P'}{P[\varphi] \xrightarrow{\varphi(\alpha)} P'[\varphi]}$$

Table 2
Structural operational rules for static operators

Definition 2.2 Two processes P, Q are observationally congruent, written $P \simeq Q$, iff:

- If $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \approx Q'$.
- If $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \stackrel{\alpha}{\Longrightarrow} P'$ and $P' \approx Q'$. ■

Both weak bisimulation and observational congruence are extended to open terms in the standard way [15,14] so to have that: two (open) terms E and F are equivalent if for every substitution of free variables for closed terms, the closed terms obtained by applying the (same) substitution to E and F are equivalent.

2.1 Completeness for BCCS with recursive definitions

2.1.1 The axioms

The standard axiomatization is composed by the axioms in tables 3, 4 and 5. As we observed in [2], an alternative equivalent smaller axiomatization can be obtained by replacing the axioms $(WUng1)$ and $(WUng2)$ with the unique axiom:

$$(WUng)\ recX.(\tau.(X+E)+F) = recX.(\tau.(E+F))$$

$(A1)$	$E+F=F+E$
$(A2)$	$(E+F)+G=E+(F+G)$
$(A3)$	$E+E=E$
$(A4)$	$E+\underline{0}=E$
$(Tau1)$	$\alpha.\tau.E=\alpha.E$
$(Tau2)$	$E+\tau.E=\tau.E$
$(Tau3)$	$\alpha.(E+\tau.F)+\alpha.F=\alpha.(E+\tau.F)$

Table 3
Axioms for BCCS

$(Unfold)$	$recX.E = E\{recX.E/X\}$	
$(Fold)$	$F=E\{F/X\} \Rightarrow F = recX.E$	X strongly guarded in E

Table 4
Unfolding and folding recursion axioms

$(FUng)$	$recX.(X+E) = recX.E$
$(WUng1)$	$recX.(\tau.X+E) = recX.(\tau.E)$
$(WUng2)$	$recX.(\tau.(X+E)+F) = recX.(\tau.X+E+F)$

Table 5
Axioms for unguarded recursion

2.1.2 Completeness for BCCS with guarded recursive definitions

Completeness for observational congruence over $BCCS_{grec}$ terms is shown by resorting to standard equation sets.

Definition 2.3 (Equation Sets) An *equation set* with *formal variables* $\tilde{X} = \{X_1, \ldots, X_n\}$ and *free variables* $\tilde{W} = \{W_1, \ldots, W_m\}$, where $\tilde{X}$ and $\tilde{W}$ are disjoint, is a set $S = \{X_i = H_i \mid 1 \le i \le n\}$ of equations such that the terms H_i $(1 \le i \le n)$ have free variables in $\tilde{X} \cup \tilde{W}$.

A *solution* of the equation set is a set of terms $E_1, \ldots, E_n$ such that: for every $1 \le i \le n$ we have that $E_i = H_i\{E_j/X_j \mid 1 \le j \le n\}$ [2] can be shown by using the axiomatization above. A single term E is said to be a *solution* of the equation set if there exists a solution $E_1, \ldots, E_n$ such that $E \equiv E_1$. ■

Definition 2.4 (Standard Equation Sets) An equation set $S = \{X_i = H_i \mid 1 \le i \le n\}$, with formal variables $\tilde{X} = \{X_1, \ldots, X_n\}$ and free variables $\tilde{W} = \{W_1, \ldots, W_m\}$, is *standard* if each term H_i $(1 \le i \le n)$ is of the form: [3]

[2] The notation used here stands for the syntactical replacement of E_j for every occurrence of X_j (for every j) inside H_i.
[3] We assume $\sum_{j \in J} E \equiv \underline{0}$ if $J = \emptyset$.

$$H_i \equiv \sum_{j \in J_i} \alpha_{i,j}.X_{f(i,j)} + \sum_{k \in K_i} W_{g(i,k)}$$
■

As in [15], for a standard equation set S, we let $X_i \xrightarrow{\alpha}_S X$ stand for "$\alpha.X$ occurs in H_i". $\xrightarrow{\alpha}_S{}^+$ denotes a non-zero length sequence of $\xrightarrow{\alpha}_S$.

Definition 2.5 A standard equation set S with formal variables $\tilde{X} = \{X_1, \ldots, X_n\}$ is *guarded* if there is no cycle $X_i \xrightarrow{\tau}_S^+ X_i$. ■

It is crucial to observe that guarded standard equation sets have *one and only one solution*. More precisely we have the following properties:

- *Existence of a solution for standard equation sets*, i.e. generalizaion of $(Unfold)$ axiom: given a standard equation set it is always possible to build a set of $BCCS_{grec}$ terms which constitutes its (canonical) solution.
- *Unicity of the solution for guarded standard equation sets*, i.e. generalizaion of $(Fold)$ axiom: different soutions of the same guarded can be shown to be equal by using the axiomatization above.

Finally, the completeness result is obtained by showing: that every $BCCS_{grec}$ term can be *represented* by a guarded standard equation set (for which the term is a solution) and that for every pair of guarded standard equation sets satisfied by observationally bisimilar $BCCS_{grec}$ terms we can build a common guarded standard equation set that is satisfied by both terms.

Proposition 2.6 (representability) *For every $BCCS_{grec}$ term E there is a guarded standard equation set that is satisfied by E.*

Proposition 2.7 (mergeability) *Let E, F be $BCCS_{grec}$ terms. If $E \simeq F$ then, given a guarded standard equation set satisfied by E and a guarded standard equation set satisfied by F, there exists a common guarded standard equation set satisfied by both E and F.*

2.1.3 Completeness Technique

The standard completeness technique is based on two steps.

- Completeness for $BCCS_{grec}$ (open) terms presented above.
- Normal form derivability: $BCCS_{grec}$ terms can always be derived as "normal" forms by using the axiomatization.

Proposition 2.8 (normal form derivability) *For every $BCCS_{rec}$ term E there exists a $BCCS_{grec}$ term F such that $E = F$.*

Normal form derivability (i.e. the proof of the proposition) involves *elimination of unguarded recursion* by using the $(FUng)$ and $(WUng)$ axioms of Table 5 (see [15]).

In general the two step schema above is used to prove completeness over (unguarded) terms of a full calculus by turning them into normal form.

2.2 Completeness for full CCS without recursive definitions

2.2.1 The axioms

The axiomatization is composed by the following axioms.

- The axioms for $BCCS$ of table 3.
- Axioms for static operators elimination (this requires the introduction of auxiliary operators like left merge and syncronization merge for parallel).

The axioms in the last item guarantee that, for each static operator, when the static operator is applied to $BCCS$ term(s) E, a $BCCS$ term can be obtained by applying the axioms.

2.2.2 Completeness Technique

The standard completeness technique is, again, based on two steps.

- Completeness over $BCCS$ terms.
- Normal form derivability: $BCCS$ terms can always be derived as "normal" forms by using the axiomatization.

Proposition 2.9 (normal form derivability) *For every CCS term P there exists a $BCCS$ term Q such that $P = Q$.*

Normal form derivability (i.e. the proof of the proposition) involves *elimination of static operators* by an *inductive structural bottom-up transformation.*

3 Finite state general processes

In this section we report about work on extending the standard weak bisimulation machinery to deal with both recursion and static operators [2,1]. For example the following observationally congruent CCS terms

$$(\,(recX.a.X) \mid (recX.\overline{a}.X)\,) \setminus \{a\} \simeq recX.\tau.X \simeq \tau.\underline{0}$$

cannot be equated by using the standard axioms presented before. The idea here is that, if we just deal with terms whose semantics is finite-state like the above one, we can still develop a complete axiomatization by extending the standard machinery.

Technically, the problem with the term above arises from the hiding behavior of the CCS parallel operator which can generate new τ actions and, as a consequence, weakly unguarded recursions. In order to solve this problem we need to express such a hiding behavior explicitly by a separate hiding operator (we use the operator E/L which hides all the actions in the set L) and to introduce the new axiom $(recX.E)/L = recX.(E/L)$ that allows us to exchange recursion and hiding, so to evaluate the effect of hiding inside serial terms E. The CCS parallel is then expressed as a combination of ACP parallel and hiding.

The focus of [2] is on producing a general process algebra such that all the operators of CCS, CSP and ACP can be expressed as a combination of the operators of such an algebra, so, e.g., in addition to ACP parallel and hiding also sequencing "$E_1; E_2$" and successful termination "$\underline{1}$" are considered (in addition to prefixing and $\underline{0}$, that now represent just deadlocked termination).

In [2] we consider the following syntactical characterization for closed terms E of the general process algebra, which guarantees finite-stateness. For any subterm $recX.F$ of E, (free) occurrences of X in F must not be in the scope of static operators (i.e. operators $E'|E''$, $E'\backslash L$, E'/L, $E'[\varphi]$) or on the left-hand side of the sequencing operator "$E; E$".

3.1 The axioms

The axiomatization is composed by the following axioms.

- The axioms for $BCCS_{rec}$ of tables 3, 4 and 5, where the "X *serial in* E" requirement is added in the folding axiom ($Fold$): the new requirement "X serial in E" makes the existing requirement "X guarded in E" well defined on the extended syntax.
- The new axiom ($RecHid$) for *recursion and hiding exchange*:

$$(RecHid)\ (recX.E)/L = recX.(E/L) \quad X \text{ serial in E}$$

- Axioms for static operator elimination (including sequencing operator "$E; E$").

The axioms in the last item are of the same kind of those considered in Section 2.2.1: they guarantee static operators (or sequencing) to be eliminated when applied to (non-recursive) BCCS terms. Concerning the general process algebra above, they guarantee that, for each static operator (including the sequencing operator), when the operator is applied to term(s) E of the form [4] $\sum_{i \in I} \alpha_i.E_i$ (or $\sum_{i \in I} \alpha_i.E_i + \underline{1}$), where E_i are unrestricted general terms, a term of the same form can be obtained by applying the axioms.

3.2 Completeness Technique

Completeness over finite-state terms belonging to the syntactical characterization above is obtained as usual (see Section 2.1.3) in two steps.

- Completeness for $BCCS_{grec}$ terms presented in Sect. 2.1.2 (successful termination can be just seen as a special new action used in $BCCS_{grec}$ terms).
- Normal form derivability: $BCCS_{grec}$ terms can always be derived from terms belonging to the syntactical characterization above as "normal" forms by using the axiomatization.

Concerning normal form derivability, terms belonging to the syntactical characterization above are shown to be transformable into normal form by com-

[4] We assume $\sum_{i \in I} E \equiv \underline{0}$ if $I = \emptyset$.

binining *unguarded recursion elimination* of $BCCS_{rec}$ (see Sect. 2.1.3) with *static operator elimination* of CCS (see Sect. 2.2.2). In particular an inductive structural bottom-up transformation of terms into normal form is performed. The inductive step consists in generating a term in normal form by applying some operator to term(s) in normal forms. This is done as follows:

- Only in the case of a *recursion* or *hiding* operator: we *remove generated unguarded recursion* via the stadard ($WUng$) and ($FUng$) axioms plus the new axiom ($RecHid$) (and other axioms for hiding). In the case of recursion, this is done just as in the standard case (Sect. 2.1.3). In case of hiding, we preliminarly propagate structurally the hiding operator top-down until leaves and then again bottom-up to the top-level to introduce new taus in the previous normal form (obtained as the effect of hiding) and then remove generated unguarded recursion.
- We then obtain a normal form by applying *static operator elimination axioms* and by *folding*: each derivative of the term (corresponding to a state of its finite-state operational semantics) is turned into a variable of a system of equations, that is expressed by recursion.

4 Refinable Actions

In this section we show how to extend the standard machinery for weak bisimulation in order to deal with refinable actions, i.e. actions which can be refined by means of an action refinement operator "$P[a \rightsquigarrow Q]$" that performs the semantic refinement of "a" executed by "P" into "Q" [6,5]. It is well known that ST bisimulation [10] is the coarsest bisimulation equivalence that is a congruence with respect to semantic action refinement. Weak ST bisimulation matches visible (non-τ) actions of concurrent systems as if they had a duration. In particular,actions are not treated as being executed atomically like in standard interleaving semantics, but the execution of an action is split into the two distinguished events of action start and action termination (pair of semi-actions). Moreover, enough information is included in semantic models so that the event of an action termination uniquely determines to which event of action start it refers to, even in the situation of auto-concurrency (i.e., multiple actions of the same type being in execution at the same time). Weak ST Bisimulation over terms of a process algebra can be expressed in several equivalent ways by using different combination of "semantics+equivalence" definitions (see [6] and the references therein). Here we consider definitions of ST semantics (taken from [6]) that allow us to obtain weak ST bisimulation by just using standard weak bisimulation on the generated special transition systems. This allows us to deal with weak ST bisimulation by a smooth extension of the standard machinery and to obtain a complete axiomatization over finite-state terms. More precisely, since the definitions of ST semantics of [6] yield finite-state transition systems for all terms which are finite-state with the standard interleaving semantics, we obtain a axiomatization which

$$a.P \xrightarrow{a_1^+} a_1^-.P \qquad a_1^-.P \xrightarrow{a_1^-} P$$

$$\frac{P \xrightarrow{a_i^+} P'}{P \,\|_{S,M}\, Q \xrightarrow{a^+_{new(M,a)}} P' \,\|_{S,upd^+(M,a,l_i)}\, Q} \; a \notin S$$

$$\frac{P \xrightarrow{a_i^-} P' \quad a\!:\!(j,l_i) \in M}{P \,\|_{S,M}\, Q \xrightarrow{a_j^-} P' \,\|_{S,upd^-(M,a,l_i)}\, Q} \; a \notin S$$

$$\frac{P \xrightarrow{a_i} P' \quad Q \xrightarrow{a_i} Q'}{P \,\|_S\, Q \xrightarrow{a_i} P' \,\|_S\, Q'} \; a \in S$$

Table 6
Structural operational rules for refinable actions

is complete for observational congruence over terms satisfying the syntactical characterization previously presented in Sect. 3.

The process algebra considered in [6] is an extension of $BCCS_{rec}$ that (apart from the action refinement operator "$P[a \rightsquigarrow Q]$") is similar to the general process algebra of Sect. 3, with the difference that the CSP parallel $E \,\|_S\, E$, where the actions in the set S are required to synchronize (while other actions are executed independently), is employed instead of ACP parallel and restriction. In particular, like in Sect. 3 we consider hiding, sequencing "$E_1; E_2$" and successful termination "$\underline{1}$" (it is well known that distinguishing between deadlocked termination and successful termination $\underline{1}$ is necessary for the semantic action refinement operator to be a congruence).

Here we consider two defintions of ST semantics from [6], corresponding to two different techniques for associating action termination events to action start events:

- *Dynamic name technique*: action termination is associated to action start by using a common *unique name* (positive integer index), generated at the moment of action start with a fixed rule.
- *Stack technique*: action termination is associated to action start by a *pointer* (positive integer index).

The *dynamic name* technique is based on the idea of dynamically assigning, during the evolution of the system, a new name to each action that starts execution, on the basis of the names assigned to the actions already started. *Names* are indexes $i \in \mathbb{N}^+$ that distinguish actions of the same type. In particular the event of starting of a visible action a is represented in semantic models by a transition labeled by a_i^+ where i is the minimum index not already used by the other actions a that have started but not terminated yet. This rule for computing indexes guarantees that names are reused and that finite models can be obtained also in the presence of recursion. The termination of the action is simply represented by a transition labeled by a_i^-, where the "identifier" i uniquely determines which action a is terminating. Since the method to compute the index for a starting action is fixed, it turns out that actions of processes that perform the same execution traces of actions get the same names. As a consequence, ST bisimilarity can simply be checked by applying standard bisimilarity to the semantic models of processes.

In order to obtain a smooth extension of standard machinery, the ST semantic models that assign indexes to semi-actions a^+ and a^- according to the *dynamic name* technique explained above must be obtained by a simple extension of standard structural operational semantics, hence in a *compositional way*. To do this, we make use of our idea of *levelwise reindexing* of actions. For obtaining compositionality, it is necessary to determine, e.g. in the case of the parallel composition operator, the computations of $P \,\|_S\, Q$ from the computations of P and Q. This is done by parameterizing in state terms each parallel operator with a mapping M. For every action $a \notin S$ started by $P \,\|_{S,M}\, Q$, M records the association between the name a_i, generated according to the fixed rule above for identifying a at the level of $P \,\|_{S,M}\, Q$, and the name a_j (which in general is different from a_i), generated according to the same rule for identifying the same action a inside P (or Q). In this way when, afterwards, such an action a terminates in P (or Q) the name a_j can be re-mapped to the correct name a_i at the level of $P \,\|_{S,M}\, Q$, by exploiting the information included in M. In M the action a of $P \,\|_{S,M}\, Q$ which gets index i is uniquely identified by expressing the unique name j it gets in P or in Q and the "location" of the process that executes it: *left* if P, *right* if Q. Such an association is, therefore, represented inside M by the triple (a, i, loc_j) with $a \in A$, indexes $i, j \in \mathbb{N}^+$ and location $loc \in Loc = \{l, r\}$, where "l" stands for left and "r" for right. In the following we use $a:(i, loc_j)$ to stand for $(a, i, loc_j) \in M$. Formally, we denote by ia an *index association*, whose elements are associations (i, loc_j). ia ranges over the set IA of partial bijections from $\mathbb{N}^+$ to $Loc \times \mathbb{N}^+$. A mapping M is a relation from A to $\mathbb{N}^+ \times (Loc \times \mathbb{N}^+)$ such that $\forall a \in A.\, M(a) \in IA$, i.e. M is a set including an independent index association for each different action type.

The non-standard structural operational semantics rules for terms are those presented in Table 6 (plus symmetrical ones, where "l_i" is replaced by "r_i"). According to what we explained above, we assume $new(M, a) = min\{k \mid k \notin dom(M(a))\}$, $upd^+(M, a, l_i) = M \cup \{a : (new(M, a), l_i)\}$ and $upd^-(M, a, l_i) = M - \{a : (j, l_i)\}$, with j the only index such that $a : (j, l_i) \in M$. Note that, for expressing the ST semantics we need to use an extended syntax for state terms that includes semi-action prefixes and the extended CSP parallel operator $P \,\|_{S,M}\, Q$, where we consider $P \,\|_{S,\emptyset}\, Q$ to stand for $P \,\|_S\, Q$ in the initial term.

The ST semantics via dynamic name presented above has all the required characteristics to be used as a smooth extension of the standard machinery to deal with refinable actions, but one: in general it does not preserve finite-stateness when the semantic action refinement operator "$P[a \rightsquigarrow Q]$" is applied (supposing both terms P and Q to be finite state). With our approach the semantics of $P[a \rightsquigarrow Q]$ can be simply defined in terms of simpler operators:

$$(P[a \leftrightarrow e] \parallel_{\{e\},\emptyset} !(e_1^+; Q; e_1^-))/\{e\}$$

where, e is a distinguished action not occurring in the set of action names $\mathcal{A}$, the bijective relabeling $\alpha \leftrightarrow \alpha'$ is defined by $\alpha \leftrightarrow \alpha' = \{(\alpha, \alpha'), (\alpha', \alpha)\} \cup \{(\alpha'', \alpha'') \mid \alpha'' \in A \cup \{e\} \wedge \alpha'' \notin \{\alpha, \alpha'\}\}$, the bang operator "$!P$" may either terminate successfully or execute "P" in such a way that a new copy of $!P$ in parallel is produced after an initial action of "P" is executed, the semantics of action start semi-actions prefixes $e_1^+.P$ is defined in the obvious way (they generate a corresponding "e_1^+" transition to reach state P), and trailing "$\underline{1}$" are omitted from prefixes, i.e. e_1^+ and e_1^- stand for $e_1^+.\underline{1}$ and $e_1^-.\underline{1}$.

The problem with finite stateness is related to the fact that the number of parallel operators generated by the bang operator grows as new actions to be refined start and terminate; therefore in the absence of an elimination rule for the terminated parallel processes (and the involved parallel operator itself) we get infinite state systems even when refining very simple recursive terms. The dynamic name technique however does not allow to introduce elimination of trailing terminated parallel processes due to "holes" in the generated set of action dynamic names (see [6]), i.e. if in a state the current set of indexes of actions a in execution (started but not terminated) is $\{1, 3\}$, we have a "hole" in the position 2.

In order to obtain finite-stateness also in the presence of the action refinement operator and to produce an axiomatization which is complete over terms that include such an operator, the more complex "stack technique" must be used, which is based on the idea of avoiding the generation of such "holes" in the sequence of started action indexes upon action termination. In particular, started actions of a given type are organized as a stack of coins over a table where the coin on the top of the stack is the action with index 1 and the other actions are indexed in increasing order from top to bottom. When a new action starts the corresponding coin is put on the top of the stack (and the old actions are renumbered accordingly). When an action terminates the corresponding coin is removed and the hole is "eliminated by gravity" (causing a renumbering of all the actions below it). Since the index of a started action change dynamically while other actions start and terminate, this technique is not based on names (seen as identifiers for actions) but is more similar to a pointer. In particular, the event of starting of an action a is represented in semantic models by a transition labeled with a^+ (so no index is observable) whilst the event of termination of an action a is represented by a transition labeled with a_i^- where i is the current position of the action on the stack.

Like the dynamic name technique, the stack techique is expressed compositionally in structural operational semantics by using the idea of levelwise reindexing. In particular the non-standard rules are those of Table 6, where:

indexes are eliminated from a^+ semi-actions (in prefixes and labels), i.e. the *new* function is no longer used; $upd^+(M, a, l_i)$ modifies M by adding $a : (1, l_1)$, by incrementing the indexes k, k' occurring inside every triple $a : (k', l_k)$ and by incrementing the index k' inside every triple $a : (k', r_k)$ for any k; and $upd^-(M, a, l_i)$ modifies M by removing $a : (j, l_i)$, with j the only index such that $a : (j, l_i) \in M$, by decrementing the indexes k, k' occurring inside every triple $a : (k', l_k)$ such that $k' > j$ (or equivalently $k > i$) and by decrementing the index k' inside every triple $a : (k', r_k)$ such that $k' > j$ (k is arbitrary).

4.1 The axioms

The axiomatization is composed of the following axioms.

- The axioms for $BCCS_{rec}$ of tables 3, 4 and 5, where α ranges over semi-actions a_i^+ (a^+ for the stack technique), a_i^- and τ and the "*X serial in E*" requirement is added in the folding axiom $(Fold)$.
- The new axiom $a.E = a_1^+.a_1^-.E$ (for the dynamic name technique) or $a.E = a^+.a_1^-.E$ (for the stack technique).
- The $(RecHid)$ axiom.
- Axioms for static operators elimination (including operator "$E; E$"): left merge and synchronization merge must be parameterized by a mapping M like in the case of parallel and in their axioms we use *compositional levelwise reindexing* like in the S.O.S. rules.
- In the case of the stack technique, the axiom that turns semantic action refinement into parallel:
  $$P[a \rightsquigarrow Q] = (P[a \leftrightarrow e] \,\|_{\{e\},\emptyset} !(e^+; Q; e_1^-))/\{e\}$$
  an axiom for turning the bang operator into a combination of left merge and recursion and an axiom for elimination of trailing terminated parallel processes.

4.2 Completeness Technique

Completeness over finite-state terms belonging to the syntactical characterization presented in Sect. 3 (where, in the case of the stack technique, also the semantic action refinement operator is considered as a static operator) is obtained as usual (see Section 2.1.3) in two steps.

- Completeness for $BCCS_{grec}$ terms presented in Sect. 2.1.2 (successful termination can be just seen as a special new action used in $BCCS_{grec}$ terms).
- Normal form derivability: $BCCS_{grec}$ terms can always be derived from terms belonging to the syntactical characterization above as "normal" forms by using the axiomatization.

Concerning normal form derivability, terms belonging to the syntactical characterization above are shown to be transformable into normal form by adopting the same procedure as that of the previous Sect. 3.2. Note that in the current context this means that during the inductive structural bottom-up transformation of terms, action are reindexed at every level as an effect of the axioms for the elimination of the parallel operator.

5 Maximal Progress

In this section we deal with the problem of extending standard process algebra with special actions σ whose execution is unprioritized with respect to standard actions [7,5,8]. As we will see, we can use such actions to represent the passage of time. In this context the precedence of standard action execution over time is often termed "maximal progress". More precisely, since we consider visible actions to represent just potential for execution (e.g. in CCS the actual execution depends on the environment executing a corresponding co-action) and we regard them to be actually executable only when they become τ, we assume just τ actions to actually preempt σ actions. In this section we will consider the basic process algebra $BCCS_{rec}$ and we will show how to adjust the standard machinery to deal with the unprioritized actions σ. Concerning the definition of syntax, we just add $\sigma \notin \mathcal{A}$ to the set of actions Act; regarding semantics of "σ" actions, we replace the standard rules for the "$+$" operator with the rule

$$\frac{E \xrightarrow{\sigma} E' \quad F \not\xrightarrow{\tau}}{E + F \xrightarrow{\sigma} E'}$$

and its symmetric. This means, i.e., that the semantic model of "$\sigma.P + \tau.Q$" is isomorphic to that of "$\tau.Q$".

The modification of the operational semantics of $BCCS_{rec}$ does not affect the congruence property of standard observational congruence (where σ is treated as a standard visible action). However, as far as producing a complete axiomatization for observational congruence over $BCCS_{rec}$ terms is concerned, the standard machinery needs to be modified. The main reason is that the axiom

$$(WUng1)\ recX.(\tau.X + E) = recX.(\tau.E)$$

is no longer sound! For instance if E is $\sigma.F$ we would obtain:

$$recX.(\tau.X + \sigma.F) \quad \not\simeq \quad \tau.\sigma.F$$

The modification to the standard machinery that is required (new set of axioms considered plus new completenes proof) is far from trivial. The basic idea is to use an auxiliay operator "$pri(E)$" to express the *scope* of priority. The semantics of $pri(P)$ is just defined by

$$\frac{P \xrightarrow{\alpha} P'}{pri(P) \xrightarrow{\alpha} P'} \qquad \alpha \neq \sigma$$

$(FUng)$	$recX.(X+E) = recX.E$	
$(WUng1)$	$recX.(\tau.X+E) = recX.(\tau.pri(E))$	
$(WUng2)$	$recX.(\tau.(X+E)+F) = recX.(\tau.X+E+F)$	
$(WUng3)$	$recX.(\tau.(pri(X)+E)+F) = recX.(\tau.X+E+F)$	

Table 7
Axioms for unguarded recursion

$(Pri1)$	$pri(\underline{0}) = \underline{0}$	
$(Pri2)$	$pri(\alpha.E) = \alpha.E$	$\alpha \neq \sigma$
$(Pri3)$	$pri(\sigma.E) = \underline{0}$	
$(Pri4)$	$pri(E+F) = pri(E)+pri(F)$	
$(Pri5)$	$pri(pri(E)) = pri(E)$	

Table 8
Axioms for the "*pri*" operator

and represents how the behavior of P is modified in the presence of an alternative initial τ transition, i.e. all initial "σ" actions (and consequent behaviors) are preempted in P. In this way the axiom $(WUng1)$ can be replaced by the new sound axiom

$$(WUng1)\ recX.(\tau.X+E) = recX.(\tau.pri(E))$$

that allows to remove weakly guarded recursion in a similar way.

5.1 The axioms

The axiomatization is composed by the following axioms.

- The axioms for $BCCS$ in table 3.
- Unfolding and folding recursion axioms in table 4 where the "X *serial in* E" requirement is added in the folding axiom $(Fold)$.
- The axiom $(MPro)$ expressing maximal-progress:

$$(MPro)\ \tau.E+F = \tau.E+pri(F)$$

- The modified axioms for unguarded recursion in Table 7 (an additional axiom "$(WUng3)$" similar to the old '$(WUng2)$" is needed to remove weakly unguarded occurrences of $pri(X)$ terms).
- The axioms for the "$pri(E)$" operator in Table 8.

5.2 Completeness for BCCS with guarded recursive definitions

In order to prove completeness for observational congruence over $BCCS_{grec}$ terms we need to introduce prioritized standard equation sets.

Definition 5.1 (Prioritized Standard Equation Sets) A standard equation set $S = \{X_i = H_i \mid 1 \leq i \leq n\}$, with formal variables $\tilde{X} = \{X_1, \ldots, X_n\}$ and free variables $\tilde{W} = \{W_1, \ldots, W_m\}$, where we assume terms H_i $(1 \leq i \leq n)$ to be represented by [5]

$$H_i \equiv \sum_{j \in J_i} \alpha_{i,j}.X_{f(i,j)} + \sum_{k \in K_i} W_{g(i,k)}$$

is *prioritized* if:

$$\exists j \in J_i : \alpha_{i,j} = \tau \;\Rightarrow\; \not\exists j \in J_i : \alpha_{i,j} = \sigma \; .$$

■

A (not involved) adaptation of the standard machinery allows us to establish one and only one solution, representability and mergeability for prioritized standard equation sets (see [7,5,8]).

5.3 Completeness Technique

Completeness over general $BCCS_{rec}$ terms is as usual obtained in two steps.

- Completeness for $BCCS_{grec}$ terms presented above.
- Normal form derivability: $BCCS_{grec}$ terms can always be derived as "normal" forms by using the axiomatization.

The proof of normal form derivability (which involves elimination of unguarded recursion via the modified "$WUng$" axioms) is the *most involved change* with respect to standard machinery (see [7,5,8]).

6 Time

In this section we present work on using unprioritized "σ" actions of the previous section to represent time in process algebra [4,8]. More precisely, we consider timed full calculi (that with respect to the basic language $BCCS_{grec}$ consider also static operators, like parallel) that preserve compatibility with the standard notion of weak bisimulation, i.e. that allow terms that do not include "σ" actions to be matched as in weak bisimulation without loosing the congruence property. In this way we are able to adapt the standard machinery for weak bisimulation to deal also with time.

In particular, we will consider two full calculi, each for a different "kind" of time: discrete real-time and Markovian stochastic time (where time is probabilistic and is expressed via exponential probability distributions). These two kinds of time (as opposed to continuous real-time and stochastic time with arbitrary probability distributions) can be expressed simply by "atomic" σ actions: in the case of discrete real-time we see σ actions as a delay of a time unit, a "tick"; in the case of Markovian stochastic time we see σ actions as

[5] We assume $\sum_{j \in J} E \equiv \underline{0}$ if $J = \emptyset$.

positive real numbers, i.e. $\sigma \in \mathbb{R}^+$, representing the parameter σ of an exponentially distributed delay (σ is also called "rate" of the distribution). The calculi that we use are based on the Hennessy-Regan approach [11] of considering time to be allowed to pass for a process, by executing "σ" transitions, only if every other process that is in parallel with it, explicitly allow time to pass via "σ" transitions as well. By taking such an approach, processes that cannot execute "σ" actions are interpreted as time deadlocked processes that block evolution of time in all the system, and, e.g., it is correct to consider $\underline{0}$ to be equivalent to $recX.\tau.X$ (that block the execution of time as well by executing an infinite sequence of prioritized τ transitions) as in standard weak bisimulation.

In the following two subsections we present the two calculi that we consider, i.e. a calculus for real-time like that in [11] and a revisitation of the calculus of Interactive Markov Chain [12] that makes it compatible with standard weak bisimulation. Both calculi are endowed with a special "timed" prefix $\alpha^t.P$ and choice $P +^t Q$ that, like in [11], allows time to evolve via execution of "σ" transitions. Such operators are different from the "standard" prefix and choice operators of $BCCS_{rec}$ that we considered in the previous Sect. 5 and here we will use the "t" superscript to distinguish them. The idea is that "timed" prefix and choice and the other operators of the timed full calculi are used to specify timed systems, i.e. the two timed full calculi are *specification level* calculi, while the basic $BCCS_{rec}$ calculus (plus an auxiliary delay hiding operator $\mathcal{H}(P)$ in the case of Markovian delays) is used to express *normal forms* for them and to produce an axiomatization. Such an axiomatization will be complete over finite-state terms belonging to a syntactical characterization (on the specification level calculus) analogous to that presented in Sect. 3.

6.1 Discrete real-time

Concerning discrete real-time, we consider the approach of [11]. For a simple calculus with prefix, summation, hiding and CSP parallel operator, in addition to the usual rules for standard action transitions we have also the rules in Table 9 for σ transitions [6]. Moreover we assume the calculus to be also endowed with a recursion $recX.P$ operator with the standard operational rule (that holds for σ transitions as well).

The transition systems obtained with the oprtaional semantics are *time deterministic*: every state either has a single outgoing σ transition or has no outgoing σ transitions.

Standard weak bisimulation (where σ is considered as a standard visible action) is a congruence with respect to all the operators above, apart summation, as expected. However, in order to obtain a congruence with respect

[6] Differently from [11] here we consider the CSP parallel operator, that allows us to keep the hiding and parallel operations expressed in a separated way. This is needed, e.g., for producing a complete axiomatization via the ($RecHid$) axiom (see Sect. 3)

$$a^t.P \xrightarrow{\sigma} a^t.P \qquad \sigma^t.P \xrightarrow{\sigma} P$$

$$\frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P +^t Q \xrightarrow{\sigma} P' +^t Q} \qquad \frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P \,\|_S\, Q \xrightarrow{\sigma} P' \,\|_S\, Q}$$

$$\frac{P \xrightarrow{\sigma} P' \qquad \not\exists a \in L.\, P \xrightarrow{a}}{P/L \xrightarrow{\sigma} P'/L}$$

Table 9
Rules for Discrete Real-Time

to all the operators we have to consider an equivalence notion that is finer than standard observational cogruence: the "root" condition of the equivalence (where standard transitions are matched as in standard observational congruence) can be left only by executing standard transitions (and not by executing σ transitions).

Definition 6.1 A relation β over processes is a discrete real-time observational bisimulation if, whenever $(P, Q) \in \beta$:

- If $P \xrightarrow{\alpha} P'$ with $\alpha \neq \sigma$ then, for some Q', $Q \xRightarrow{\alpha} Q'$ and $P' \approx Q'$.
- If $Q \xrightarrow{\alpha} Q'$ with $\alpha \neq \sigma$ then, for some P', $P \xRightarrow{\alpha} P'$ and $P' \approx Q'$.
- If $P \xrightarrow{\sigma} P'$ then, for some Q', $Q \xrightarrow{\sigma} Q'$ and $(P', Q') \in \beta$.
- If $Q \xrightarrow{\sigma} Q'$ then, for some P', $P \xrightarrow{\sigma} P'$ and $(P', Q') \in \beta$.

Two processes P, Q are discrete real-time observationally congruent, written $P \simeq_{DRT} Q$, iff (P, Q) is included in some discrete real-time observational bisimulation. ∎

6.2 Markovian stochastic time

Concerning Markovian stochastic time, we consider Revisited Interactive Markov Chains [4]. For a simple calculus with prefix, summation, hiding and CSP parallel operator, in addition to the usual rules for standard action transitions we have also the rules in Table 9 for $\sigma \in \mathbb{R}^+$ transitions. In the operational rule for prefix, $\tilde{\sigma}$ is any rate: since, as we will see, equivalence over terms abstract from the rate of selfloops (transitions going from one state to itself) of σ transitions the particular choice of $\tilde{\sigma} \in \mathbb{R}^+$ is not important. Note that, the interleaved execution of σ actions in parallel and choice is justified by the Markov property of exponential probability distributions, and correctly represents independent passage of time both in P and Q. Moreover we assume the calculus to be also endowed with a recursion $recX.P$ operator with the standard operational rule (that holds for σ transitions as well).

The equivalence that we consider is a simple stochastic extension of weak bisimulation, called weak Markovian Bisimulation [4], where equivalent terms

P and Q are required to have matching weak transitions of σ actions as for standard visible actions and we also have a quantitative requirement about the values of σ of matching weak transitions that is derived from (selfloop insensitive) Markovian bisimulation: sums of alternative σ moves labeling weak transitions that reach same equivalence classes (excluded the equivalence class that includes P and Q themselves) must match.

The selfloop insensitive variant of Markovian bisimulation that we introduced in [4] is, in turn, a coarser variant of standard strong Markovian bisimulation [13] (which instead considers also the equivalence class that includes P and Q themselves) that preserves the steady state behaviour of the underlying Continuous Time Markov Chains. The transient behavior and the performance evaluation of the model in general is not changed, provided that the termination events of individual timed σ actions are considered to be *unobservable*, e.g. successive execution of two timed actions is not distinguished from the execution of a single corresponding timed action: the equivalence just guarantee correspondance in terms of the probabilistic distribution of the overall amount of time passed.

Weak Markovian bisimulation is a congruence with respect to all the operators above, apart summation, as expected. In order to obtain a congruence with respect to all the operators we have to introduce a "root" condition in a similar way as for the discrete real-time case: the "root" of the equivalence, where standard transitions are matched as in standard observational congruence and σ actions are matched as in (strong) selfloop insensitive markovian Bisimulation, can be left only by executing standard transitions (and not by executing σ transitions). We call this equivalence Markovian stochastic time observational congruence, written $\simeq_{MST}$ (see [4] for a precise definition).

We would like to note explicitely that, the possibility of adopting a coarser equivalent notion with respect to standard Markovian Bisimulation, where markovian delays are considered as *unobservable*, is a consequence of the adoption of the time choice operator "$P +^t Q$" instead of the standard operator "$P + Q$" of the basic $BCCS_{rec}$ calculus (and of all the other Markovian calculi like e.g. [12]): while in "$P + Q$", where σ actions resolve the choice, an initial selfloop of σ actions inside P or Q becomes a σ transition that changes the system state, this does not happen in "$P +^t Q$". Technically, this allows "$P +^t Q$" to be a congruence with respect to "$\simeq_{MST}$" that is selfloop insensitive. The adoption of a selfloop insensitive equivalence besides being convenient (in that is coarser) is also adequate for the definition of the semantics of operator "$\alpha^t.P$" where we do not want the actual system behavior to be dependent on the particular $\tilde{\sigma}$ that is chosen.

6.3 The axioms

Producing a complete axiomatization for the two full timed calculi requires them to be extended to encompass operators of the basic $BCCS_{rec}$ calculus

$$a^t.P \xrightarrow{\tilde{\sigma}} a^t.P \qquad\qquad \sigma^t.P \xrightarrow{\sigma} P$$

$$\frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma'}}{P +^t Q \xrightarrow{\sigma} P' +^t Q} \qquad \frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma'}}{P \,\|_S\, Q \xrightarrow{\sigma} P' \,\|_S\, Q}$$

$$\frac{P \xrightarrow{\sigma} P' \qquad \not\exists a \in L.\, P \xrightarrow{a}}{P/L \xrightarrow{\sigma} P'/L}$$

Table 10
Rules for Exponentially Timed Moves

so that we can express normal forms of terms.

More precisely, in the case of the discrete time calculus, we use as terms for normal forms the terms of the basic $BCCS_{rec}$ calculus whose semantics is time deterministic.

In the case of the Markovian stochastic time calculus, we consider *observable* Markovian delays, denoted by σ^o (to distinguish them from *unobservable* ones used in the specification calculus) to be used in normal forms: in the $BCCS_{rec}$ syntax and in the rules of Table 1 we consider α to range over standard actions in $\mathcal{A} \cup \{\tau\}$ plus σ^o actions (with $\sigma \in \mathbb{R}^+$), instead of σ actions. Markovian stochastic time observational congruence is consistently extended to deal with observable Markovian delays σ^o as well: in equivalent terms P and Q, σ^o (weak) transitions are matched like σ (weak) transitions, with the only difference that the target selflooping equivalence class that includes P and Q themselves is not excluded when matching sums of σ^o rates, as in standard Markovian bisimulation [13]. Normal forms also make use of a delay hiding operator $\mathcal{H}(P)$ defined by:

$$\frac{P \xrightarrow{\alpha} P'}{\mathcal{H}(P) \xrightarrow{\alpha} P'} \;\; \alpha \in \mathcal{A} \cup \{\tau\} \qquad\qquad \frac{P \xrightarrow{\sigma^o} P'}{\mathcal{H}(P) \xrightarrow{\sigma} P'}$$

Intuitively, $\mathcal{H}(P)$ turns every observable exponential action that is executed by P as an initial action into an unobservable one with the same rate. Terms used as normal forms belong to the following basic calculus two-level syntax:

$$E ::= \underline{0} \mid X \mid \mathcal{H}(S) \mid recX.E$$
$$S ::= S + S \mid a.E \mid \tau.E \mid \sigma^o.E$$

The idea is that, by exploiting the delay hiding operator $\mathcal{H}(P)$, we can express terms of the Markovian stochastic calculus, which use *unobservable* Markovian delays, in terms of normal forms which use *observable* Markovian delays and the standard "+" operator which is compatible with them.

The axiomatization for the discrete real-time calculus is composed of the following axioms.

- The axioms in Sect. 5.1 of the axiomatization of $BCCS_{rec}$ with maximal progress where:
  - The "X *serial in* E" requirement is added in the folding axiom $(Fold)$.
  - The axioms $(A3)$ and $(FUng)$ expressing idempotency of "+" are replaced by the axioms

$$\begin{array}{ll} (A3) & E + pri(E) = E \\ (FUng) & recX.(X + pri(E)) = recX.pri(E) \end{array}$$

    This is due to *time determinism*, i.e. in $P + Q$, σ actions cannot be executed by both P and Q, so full idempotency is not needed.
  - The $(Tau1-3)$ axioms are restricted to $\alpha \in \mathcal{A} \cup \{\tau\}$ instead of a general α action (that would include also timed actions).
  - We consider an additional τ elimination axiom for time:

    $(Tau4) \quad \alpha.F\{\sigma.\tau.E/X\} = \alpha.F\{\sigma.E/X\} \qquad \alpha\in\mathcal{A}\cup\{\tau\}$ and X serial in F

    This axiom is a restricted version of old axiom $(Tau1)$ of the axiomatization of $BCCS_{rec}$ with maximal progress in the case $\alpha = \sigma$, that now we have excluded. The restriction accounts for the new extended root condition of equivalence.
- The $(RecHid)$ axiom.
- Axioms for static operators elimination. Time prefix "$\alpha^t.E$" and choice "$E +^t E$" are treated like static operators. In the case $\alpha \neq \sigma$, "$\alpha^t.E$" is eliminated by the single axiom (that exploits maximal progress of"+"):

$$(TPre1) \quad \alpha^t.E = recX.(\sigma.X + \alpha.E) \qquad \alpha\in\mathcal{A}\cup\{\tau\}$$

  in the case $\alpha = \sigma$ by the trivial axiom, called $(TPre2)$, "$\sigma^t.E = \sigma.E$". "$E +^t E$" is eliminated by using auxiliary operators with a similar role as left merge and synchronization merge for the parallel operator.

The axiomatization for the Markovian stochastic time calculus is composed of the following axioms.

- The same modification to the axioms in Sect. 5.1 of the axiomatization of $BCCS_{rec}$ with maximal progress considered for the discrete real-time calculus (with the only difference that, due to the structure of the basic calculus used for normal forms described above, terms used in axioms are also endowed with delay hiding operators $\mathcal{H}(P)$, σ^o actions are used instead of σ actions, and the definition of serial variable is extended to also admit the variable to be in the scope of operators $\mathcal{H}(P)$) plus the following two axioms:

$$\begin{array}{ll} (Exp) & \sigma_1^o.E + \sigma_2^o.E = (\sigma_1^o + \sigma_2^o).E \\ (ExpRec) & recX.\mathcal{H}(\sigma_1^o.X + \sigma_2^o.E + F) = recX.\mathcal{H}(\sigma_2^o.E + F) \end{array}$$

  Axiom (Exp) characterizes *stochastic delay summation* performed by standard Markovian bisimulation. Axiom $(ExpRec)$ characterizes the self-loop insensitivity property of our coarser notion of Markovian bisimulation.

Note that, while in the context of discrete real-time the modification of axioms $(A3)$ and $(FUng)$ expressing idempotency of "+" is due to the time determinism property, here the same modification is due to stochastic delay summation property that causes non-idempotency of "+" over timed actions.

- The $(RecHid)$ axiom.
- Axioms for static operators elimination. These axioms are different from the discrete real-time case due to the different treatment of timed actions (here they are interleaved instead of being synchronized). Included in the set of axioms provided for the elimination of every static operator here we must consider also an axiom for the operator $\mathcal{H}(P)$, besides the other operators of $BCCS$. For timed prefix "$\alpha^t.E$" we have the single axiom

$$(TPre) \quad \alpha^t.E = recX.\mathcal{H}(\sigma^o.X + a.E)$$

that corresponds to the axioms $(TPre1)$ and $(TPre2)$ presented for the discrete real-time calculus. $(TPre2)$ is obtained from $(TPre)$ by using the selfloop insensitivity axiom $(ExpRec)$. Note that here the positive real value of σ^o is universally quantified: any choice is acceptable due to selfloop insensitivity.

6.4 Completeness for timed basic terms with guarded recursive definitions

In order to prove completeness of the axiomatization for timed observational congruence over basic terms to be used as normal forms (defined in the previous Sect. 6.3), in the case of guarded recursive definitions, we make use of *timed prioritized standard equation sets.*

Timed prioritized standard equation sets are defined as a variant of prioritized standard equation sets (presented in Sect. 5.2) where (non-free) *variables are classified in two types* depending if they are "reached" from the initial variable by just executing timed actions or not (i.e. if they are involved in the root condition of equivalence applied to the terms represented by the equation sets or not).

More precisely, for discrete real-time we have in addition a constraint related to time determinism: in every equation there can be at most one σ action. For Markovian stochastic time we have in addition that sums of actions in equations are enclosed into a delay hiding operator $\mathcal{H}(P)$ and that timed actions in the equations are observable σ^o actions.

A (not involved) adaptation of the standard machinery allows us to establish one and only one solution and representability for timed prioritized standard equation sets. Concerning the proof of mergeability, the standard machinery must be significantly modified. We need to resort to:

- Preliminar "saturation" of τ actions after σ prefixes in basic terms: given two equivalent terms we have to add τ prefixes after σ prefixes not involved in the root condition (by using the ($Tau4$) axiom) before representing them with timed prioritized equation sets.

- Development of a simplified equivalence implied by weak bisimulation to be used when merging timed prioritized standard equation sets:
  · We start by considering an implied simplified form of weak bisimulation definition where $\stackrel{\sigma}{\Longrightarrow}$ transitions of a term Q to be matched to $\stackrel{\sigma}{\longrightarrow}$ transitions of a term P are replaced by $\stackrel{\sigma}{\longrightarrow}\ \stackrel{\tau}{\longrightarrow}^*$ transitions and the check is performed only if Q cannot execute τ transitions: this can be done due to maximal progress (i.e. priority of τ over σ). Such a simplified form comes from the machinery related to the $BCCS_{rec}$ calculus with maximal progress presented in Sect. 5.
  · We then show that we can further turn $\stackrel{\sigma}{\longrightarrow}\ \stackrel{\tau}{\longrightarrow}^*$ transitions into strong $\stackrel{\sigma}{\longrightarrow}$ moves, due to discrete time determinism (for the discrete real-time calculus) or stochastic delay summation (for the stochastic Markovian time calculus). This justifies the absence of a correspondent of axiom ($Tau3$) for timed actions (case $\alpha = \sigma$ in ($Tau3$) axiom).

6.5 Completeness Technique

Completeness over finite-state terms belonging to the syntactical characterization presented in Sect. 3 (where, in the case of the "$P +^t Q$" operator, free occurrences of X are required to occur in P and Q guarded by a standard action, i.e. in the scope of an "$\alpha^t.P$" prefix with $\alpha \in \mathcal{A} \cup \{\tau\}$) is obtained as usual (see Section 2.1.3) in two steps.

- Completeness for timed basic terms with guarded recursive definitions, presented in Sect. 6.4
- Normal form derivability: timed basic terms with guarded recursive definitions can always be derived from terms belonging to the syntactical characterization above as "normal" forms by using the axiomatization.

Concerning normal form derivability, terms belonging to the syntactical characterization above are shown to be transformable into normal form by adopting the same procedure as that of the previous Sect. 3.2 that combines uguarded recursion elimination of basic terms with static operator elimination by performing an inductive bottom-up transformation of terms. More precisely, here uguarded recursion elimination in basic terms is performed as for the calculus $BCCS_{rec}$ with maximal progress (see Sect. 5.3) with a minor modification due to non general idempotency of "$P + Q$" operator.

7 Conclusion

The most direct consequence of the work presented in this paper is the development of the extension of the standard machinery for weak bisimulation to deal with calculi for *continuous real-time* and *general Stochastic time*: this can be done by combining the calculus with refinable ST actions with the calculs with (maximal-progress and) time, by considering unprioritized σ actions as ST actions, similarly to what we did in [9,5]. Note that, as shown by the machinery in [5], usage of ST actions in the context of general Stochastic time allows, not only to correctly represent systems with generally distributed delays, but also to support their refinement with terms made up of exponentially distributed delays (approximation via phase-type distributions). Finally, future work could be done in extending the completeness results to *all* finite-states term (not only terms of the considered syntactical characterization, i.e. static operators are not replicable by recursion): this would lead to a corresponding enlargement of the completeness result for any of the considered calculi.

References

[1] J.C.M. Baeten, M. Bravetti *"A generic process algebra"*, in Proc. of the meeting *Algebraic Process Calculi: The First Twenty Five Years and Beyond (PA'05)*, ENTCS 162:65-71, Bertinoro (Italy), August 2005

[2] J.C.M. Baeten, M. Bravetti *"A Ground-Complete Axiomatization of Finite State Processes in Process Algebra"*, in Proc. of the *16th Int. Conf. on Concurrency Theory (CONCUR'05)*, LNCS 3653:248-262, San Francisco (CA, USA), August 2005

[3] J.C.M. Baeten and M. Bravetti *"A ground-complete axiomatization of finite state processes in process algebra"*, Technical Report CS Report 05-18, Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 2005.

[4] M. Bravetti, *"Revisiting Interactive Markov Chains"*, in Proc. of the *3rd Int. Workshop on Models for Time-Critical Systems (MTCS 2002)*, ENTCS 68(5), Brno (Czech Republic), August 2002

[5] M. Bravetti, *"Specification and Analysis of Stochastic Real-Time Systems"*, PhD Thesis, University of Bologna (Italy), February 2002

[6] M. Bravetti, R. Gorrieri, *"Deciding and Axiomatizing Weak ST Bisimulation for a Process Algebra with Recursion and Action Refinement"*, in ACM Transactions on Computational Logic 3(4):465-520, 2002

[7] M. Bravetti, R. Gorrieri, *"A Complete Axiomatization for Observational Congruence of Prioritized Finite-State Behaviors"*, in Proc. of the *27th Int. Colloquium on Automata, Languages and Programming (ICALP 2000)*, U. Montanari, J.D.P. Rolim and E. Welzl editors, LNCS 1853:744-755, Geneva (Switzerland), July 2000

[8] M. Bravetti, R. Gorrieri, *"A Uniform Approach for Expressing Time in Process Algebra"*, submitted to Theoretical Computer Science.

[9] M. Bravetti, R. Gorrieri, *"The Theory of Interactive Generalized Semi-Markov Processes"*, in Theoretical Computer Science 282(1):5-32, 2002

[10] R.J. van Glabbeek, F.W. Vaandrager, *"Petri Net Models for Algebraic Theories of Concurrency"*, in Proc. of the *Conf. on Parallel Architectures and Languages Europe (PARLE '87)*, LNCS 259:224-242, Eindhoven (The Netherlands), 1987

[11] M. Hennessy, T. Regan, *"A Process Algebra for Timed Systems"*, in Information and Computation, 117(2):221-239, 1995

[12] H. Hermanns, *"Interactive Markov Chains"*, Ph.D. Thesis, Universität Erlangen-Nürnberg (Germany), 1998

[13] J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996

[14] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989.

[15] R. Milner, *"A complete axiomatization for observational congruence of finite-state behaviours"*, in Information and Computation 81:227-247, 1989