A general conservative extension theorem in process algebra

C. Verhoef [a*]

[a] Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
email chrisv@win.tue.nl

We prove a general conservative extension theorem for transition system based process theories with easy-to-check and reasonable conditions. The core of this result is another general theorem giving sufficient conditions for a system of operational rules and an extension ensuring that provable transitions from an original term in the extension are the same as in the original system. As a simple corollary of the conservative extension theorem we prove a completeness theorem. As a first application of our conservativity results, we prove a general theorem giving sufficient conditions to reduce the question of ground confluence modulo some equations for a large term rewriting system associated with an equational process theory to a small term rewriting system under the condition that the large system is a conservative extension of the small one. We provide many other applications to show that our results are useful. The applications include (but are not limited to) various real and discrete time settings in CCS, ACP, and ATP and the notions projection, renaming, state operator, priority, recursion, the silent step (both the weak and branching variants), and the empty process (or combinations of these notions).
Keyword Codes: F.1.2, F.3.2, F.4.3.
Keywords: Concurrency; Operational Semantics; Algebraic language theory.

1. INTRODUCTION

In the past few years people working in the area of process algebra have started to extend process theories such as CCS, CSP, and ACP with, for instance, real-time or probabilistics. A natural question that arises is whether or not such an extension is somehow related with its subtheory, for instance, whether or not the extension is conservative in some sense. If we add new operators and/or rules to a particular transition system it would be nice to know whether or not provable transitions of a term in the original system are the same as those in the extended system for that term; we will call this property *operational conservativity* (cf. [1]). Or, if we extend an axiomatical framework with new operators and/or equations it would be interesting to know whether or not an equality in the extended framework between two original closed terms can also be derived in the original framework. When no new identities between closed terms in the original frame-

*Full support received from the European Communities under CONCUR 2, BRA 7166.

work are provable from the extension, we call the extension an *equational conservative* extension. This is a well-known property under the name of conservativity; we just added the adjective 'equational' to prevent possible confusion.

A frequently used method to prove that an equational theory is a conservative extension of a subtheory is to perform a term rewriting analysis. In process algebra such an analysis is often very complex because the rewriting system associated with a process algebra seems to need in an essential way term rewriting techniques modulo the equations without a clear direction (such as commutativity of the choice). Moreover, these term rewriting systems generally have no "nice" properties making a term rewriting analysis a simple tool for conservativity. We mention that such term rewriting systems are not regular, which implies that confluence (modulo some equations) is not straightforward and we mention that the term rewriting relation induced by the rewrite rules is not commuting with the equality induced by the equations without a direction, which means that termination modulo these equations is not at all easy to prove. Let us briefly mention two examples to make the problems a bit more concrete. Bergstra and Klop [2] mention that for the confluence modulo some equations of their term rewriting system, they need to check ± 400 cases (which they left to the reader as an exercise). Jouannaud communicated to us that, in general, it is very hard (and unreliable) to make such exercises by hand but they can possibly be checked by computer. Our second example originates from Akkerman and Baeten [3]. They show that a fragment of ACP with the branching τ is both terminating and confluent modulo associativity and commutativity of the alternative composition. Akkerman told us that it is not clear to him how this result could also be established for the whole system and thus yielding a conservativity result. However, according to Baeten it is not a problem to establish these results; needless to say that their term rewriting analysis is rather complicated.

To bypass the abovementioned problems involving term rewriting, we propose an alternative method to prove conservativity. We prove a general theorem with reasonable and easy-to-check conditions giving us immediately the operational and equational conservativity in many cases. For instance, with our results, the conservativity of the above mentioned systems with problematic term rewriting properties is peanuts. The idea is that we transfer the question of equational conservativity to that of operational conservativity rather than to perform a term rewriting analysis. The only thing that remains to be done in order to prove the operational conservativity is to check our simple conditions for the operational rules. For the equational conservativity we moreover demand completeness for the subtheory and soundness for its extension. These conditions are in our opinion reasonable, because relations between equational theories only become important if the theories themselves satisfy well-established basic requirements. Moreover, our result works for a large class of theories, which is certainly not the case with a term rewriting analysis. All this implies that we give a semantical proof of conservativity, which might be seen as a drawback since a term rewriting analysis often is model independent (but see Bergstra and Klop [4] for a semantical term rewriting analysis). However, since the paper of Plotkin [5], the use of labelled transition systems as a model for operational semantics of process theories is widespread; so virtually every process theory has an operational semantics of this kind. Moreover, our equational conservativity result holds for all semantical equivalences that are definable exclusively in terms of transition relations. We recall that the following semantical equivalences are examples of such equivalences: trace equivalence, completed trace equivalence, failure equivalence, readiness equivalence, failure trace equivalence, ready trace equivalence, possible-future equivalence, simulation equivalence, complete simulation equivalence, ready simulation equivalence, nested simulation equivalence, strong bisimulation equivalence, weak bisimulation equivalence, η bisimulation equivalence, delay bisimulation equivalence, branching bisimulation equivalence, and more equivalences. We refer to Van Glabbeek's linear time – branching time spectra [6] and [7] for more information on these equivalences. In [6] and [7], references to the origins (and their use) of these semantics can be found.

As a result we now can prove conservativity without using the confluence property. However, it is widely recognized that confluence itself is an important property, for instance, for computational or implementational purposes. So, at this point the question arises: "Why bother about such a general conservative extension theorem if we still have to prove confluence for each particular system and get the conservativity as a by-product?" The answer is that once we have the conservativity we can considerably reduce the complexity of the ground confluence as a by-product. We prove a general reduction theorem stating that in many cases a conservative extension is ground Church-Rosser modulo some equations if the basic system already has this property. For instance, the 400 cases of Bergstra and Klop [2] reduce to a term rewriting analysis with only five rewrite rules and two equations. We should note, however, that they prove (modulo 400 cases!) the confluence for open terms (although they only need the closed case), whereas our reduction theorem gives the closed case in their situation. In fact, we show that conservativity and ground Church-Rosser are, in some sense, equally expressive properties.

Another advantage of our approach is that it also works for process algebras with really bad term rewriting properties, such as process algebras containing the three τ laws of Milner, where the term rewriting approach breaks down; see, e.g., [4]. We will treat these examples in this paper.

Now that we have given some motivation for this paper we discuss its organization.

In section 2 we recall some general SOS definitions of Verhoef [8]. We will provide a running example to elucidate the abstract notions. In section 3 we formally define the notions of operational and equational conservativity. Then we prove a general operational conservativity theorem, a general equational conservativity theorem and a simple corollary concerning completeness. Also here we provide our running example. In the next section we will recall some basic term rewriting terminology to prove the abovementioned reduction theorem on the ground Church-Rosser property modulo some equations. In section 5 we will give the reader an idea of the applicability of our general theorems. Surprisingly, we could not find any conservativity result in the literature for which our conservativity theorem could not be applied, as well. The last section contains concluding remarks and briefly discusses possible future work.

1.1. Related work

In this subsection we briefly mention related work. Nicollin and Sifakis [9] prove conservativity—in some particular cases—using the same general approach as we propose in this paper, namely a semantical approach. We will discuss their conservativity results (and new results) in section 5. The notion that we call in this paper operational conservativity originates from Groote and Vaandrager [1] under the name conservativity. In Groote [10] and in Bol and Groote [11] this notion also appears. In all these papers this notion is used for a different purpose than ours. Aceto, Bloom and Vaandrager [12] introduce a so-called disjoint extension, which is a more restricted form of an operational conservative extension; they need this restriction for technical reasons. They present an algorithm generating a sound and complete axiomatization if the operational rules satisfy certain criteria. Bosscher [13] studied term rewriting properties of such axiomatizations by looking at the form of the operational rules.

2. SOME GENERAL SOS DEFINITIONS

In this section we briefly recall some notions concerning general SOS theory that we will need later on in the next section. We follow Verhoef [8] since at the moment this treatment is the most general one available. To elucidate the formal notions we intersperse them with a running example.

We assume that we have an infinite set V of variables with typical elements $x, y, z, \ldots$. A (single sorted) signature Σ is a set of function symbols together with their arity. If the arity of a function symbol $f \in \Sigma$ is zero we say that f is a constant symbol. The notion of a term (over Σ) is defined as expected: $x \in V$ is a term; if $t_1, \ldots, t_n$ are terms and if $f \in \Sigma$ is n-ary then $f(t_1, \ldots, t_n)$ is a term. A term is also called an open term; if it contains no variables we call it closed. We denote the set of closed terms by $C(\Sigma)$ and the set of (open) terms by $O(\Sigma)$. We also want to speak about variables occurring in terms: let $t \in O(\Sigma)$ then $var(t) \subseteq V$ is the set of variables occurring in t.

A substitution σ is a map from the set of variables into the set of terms over a given signature. This map can easily be extended to the set of all terms by substituting for each variable occurring in an open term its σ-image.

Definition 2.1 *A* term deduction system *is a structure* (Σ, D) *with* Σ *a signature and* D *a set of* deduction rules. *The set* $D = D(T_p, T_r)$ *is parameterized with two sets, which are called (following usual process algebra terminology) respectively the set of* predicate symbols *and the set of* relation symbols. *Let* $s, t,$ *and* $u \in O(\Sigma)$, $P \in T_p$, *and* $R \in T_r$. *We call expressions* $Ps, \neg Ps$, tRu, *and* $t\neg R$ formulas. *We call the formulas* Ps *and* tRu positive *and* $\neg Ps$ *and* $t\neg R$ negative. *If* S *is a set of formulas we write* $\mathrm{PF}(S)$ *for the subset of positive formulas of* S *and* $\mathrm{NF}(S)$ *for the subset of negative formulas of* S.

A deduction rule $d \in D$ *has the form*

$$\frac{H}{C}$$

with H *a set of formulas and* C *a positive formula; to save space we will also use the notation* H/C. *We call the elements of* H *the hypotheses of* d *and we call the formula* C *the* conclusion *of* d. *If the set of hypotheses of a deduction rule is empty we call such a rule an* axiom. *We denote an axiom simply by its conclusion provided that no confusion can arise. The notions "substitution", "var", and "closed" extend to formulas and deduction rules as expected. Note that the overload of the symbol* C *in* $C(\Sigma)$ *and* H/C *is harmless.*

Table 1: BPA with discrete time.

$a \xrightarrow{a} \surd$	$\dfrac{x \xrightarrow{a} x'}{xy \xrightarrow{a} x'y}$	$\dfrac{x \xrightarrow{a} \surd}{xy \xrightarrow{a} y}$
$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x' \xleftarrow{a} y + x}$	$\dfrac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd \xleftarrow{a} y + x}$	$\sigma_d(x) \xrightarrow{\sigma} x$
$\dfrac{x \xrightarrow{\sigma} x'}{xy \xrightarrow{\sigma} x'y}$	$\dfrac{x \xrightarrow{\sigma} x', y \xrightarrow{\sigma} y'}{x + y \xrightarrow{\sigma} x' + y'}$	$\dfrac{x \xrightarrow{\sigma} x', y \not\xrightarrow{\sigma}}{x + y \xrightarrow{\sigma} x' \xleftarrow{\sigma} y + x}$

Remark 2.2 *Note that arbitrary many premises are allowed in the set of hypotheses of a deduction rule. This generality is useful in real-time process algebras where it is very natural to have continuously many premises. See, for instance, Klusener [14] for such operational rules.*

Example 2.3 *As a running example we present an operational semantics that originates from Baeten and Bergstra [15] of a basic process language with relative discrete time: BPA*$_{\mathrm{dt}}$. *Here, we introduce it as an example of a term deduction system. The signature contains constants a (a in the current time slice) for each $a \in A$ (A is the set of atomic actions and is a parameter of the theory), alternative and sequential composition, and a unary operator σ_d, called the discrete time unit delay. It is not hard to see that the above signature plus the semantics in table 1 is an example of a term deduction system. We have relations $\cdot \xrightarrow{a} \cdot$ for all $a \in A$, a relation $\cdot \xrightarrow{\sigma} \cdot$ with $\sigma \notin A$ and predicates $\cdot \xrightarrow{a} \surd$ for all $a \in A$. The intended interpretation of $x \xrightarrow{a} x'$ is that a process x evolves into x' when executing atomic action a. The meaning of $x \xrightarrow{a} \surd$ is that process x terminates successfully after the execution of a. With $x \xrightarrow{\sigma} x'$ we mean that x evolves into x' by moving to the next time slice.*

Definition 2.4 *Let T be a term deduction system. Let $F(T)$ be the set of all closed formulas over T. We denote the set of all positive formulas over T by* $\mathrm{PF}(T)$ *and the negative formulas by* $\mathrm{NF}(T)$. *Let $X \subseteq \mathrm{PF}(T)$. We define when a formula $\varphi \in F(T)$* holds *in X; notation $X \vdash \varphi$.*

$$
\begin{array}{lll}
X \vdash sRt & \textit{if} & sRt \in X, \\
X \vdash Ps & \textit{if} & Ps \in X, \\
X \vdash s\neg R & \textit{if} & \forall t \in C(\Sigma) : sRt \notin X, \\
X \vdash \neg Ps & \textit{if} & Ps \notin X.
\end{array}
$$

The purpose of a term deduction system is to define a set of positive formulas that can be deduced using the deduction rules. For instance, if the term deduction system is a transition system specification then a transition relation is such a set. For term deduction systems without negative formulas this set comprises all the formulas that can be proved by a well-founded proof tree. If we allow negative formulas in the premises of a deduction rule it is no longer obvious which set of positive formulas can be deduced using the deduction rules. Bloom, Istrail, and Meyer [16] formulate that a transition relation must agree with a transition system specification. We will use their notion; it is only adapted to be able to incorporate predicates.

Definition 2.5 *Let $T = (\Sigma, D)$ be a term deduction system and let $X \subseteq \mathrm{PF}(T)$ be a set of positive closed formulas. We say that X* agrees with T *if a formula $\varphi \in X$ if and only if there is a deduction rule instantiated with a closed substitution such that the instantiated conclusion equals φ and all the instantiated hypotheses hold in X. More formally: X agrees with T if*

$$\varphi \in X \iff \exists\, H/C \in D, \sigma : V \longrightarrow C(\Sigma) : \sigma(C) = \varphi, \forall h \in H : X \vdash \sigma(h).$$

Definition 2.6 *Let $T = (\Sigma, D)$ be a term deduction system. A mapping $S : \mathrm{PF}(T) \longrightarrow \alpha$ for an ordinal α is called a* stratification *for T if for all deduction rules $H/C \in D$ and closed substitutions σ the following conditions hold. For all $h \in \mathrm{PF}(H)$ we have $S(\sigma(h)) \leq S(\sigma(C))$; for all $s\neg R \in \mathrm{NF}(H)$ we have for all $t \in C(\Sigma) : S(\sigma(sRt)) < S(\sigma(C))$; for all $\neg Ps \in \mathrm{NF}(H)$ we have $S(\sigma(Ps)) < S(\sigma(C))$. We call a term deduction system* stratifiable *if there exists a stratification for it.*

Example 2.7 *We will give a rule of thumb for finding a stratification and apply this to our running example. In most cases we can find a stratification by measuring the complexity of a positive formula in terms of counting a particular symbol occurring in the conclusion of a rule with negative antecedents. In our case there is just one rule with a negative antecedent. In its conclusion we find the function symbol $+$. Let t be a closed term with n occurrences of this symbol. Then the map $S(t \xrightarrow{\sigma} t') = n$ is a stratification (t' is a closed term).*

Next, we assign to a term deduction system a (regular) ordinal expressing a uniform upper bound for the number of premises in the deduction rules. We use this upper bound for proof-technical reasons. (See Klusener [14] for rules with infinitely many premises.)

Definition 2.8 *Let V be a set. If $0 \leq |V| < \aleph_0$ we define the* degree *of V, denoted $d(V)$, to be ω_0. If $|V| = \aleph_\alpha$ for an ordinal $\alpha \geq 0$, we define $d(V) = \omega_{\alpha+1}$.*

Let $T = (\Sigma, D)$ be a term deduction system. The degree $d(H/C)$ of a deduction rule $H/C \in D$ is the degree of its set of positive premises: $d(H/C) = d(\mathrm{PF}(H))$. Let $\omega_\alpha = \sup\{d(H/C) : H/C \in D\}$. The degree $d(T)$ of a term deduction system T is ω_0 if $\alpha = 0$ and $\omega_{\alpha+1}$ otherwise.

Example 2.9 *It is easy to see that the degree of our running example is ω_0.*

Next, we will define a set of positive formulas from which we will show that it agrees with a given term deduction system.

Definition 2.10 *Let $T = (\Sigma, D)$ be a term deduction system and let $S : \mathrm{PF}(T) \longrightarrow \alpha$ be a stratification for an ordinal number α. We define a set $T_S \subseteq \mathrm{PF}(T)$ as follows.*

$$T_S = \bigcup_{i<\alpha} T_i^S, \quad T_i^S = \bigcup_{j<d(T)} T_{i,j}^S.$$

We will need unions over T_i^S and $T_{i,j}^S$ in proofs; so, we introduce the following notations

$$U_i^S = \bigcup_{i'<i} T_{i'}^S \ (i \leq \alpha), \quad U_{i,j}^S = \bigcup_{j'<j} T_{i,j'}^S \ (j \leq d(T)).$$

Provided no confusion arises, we drop the sub- and superscripts S and, for instance, render U_i^S as U_i and $T_S \vdash \varphi$ as $T \vdash \varphi$. Now we define for all $i < \alpha$ and for all $j < d(T)$ the set $T_{i,j} = T_{i,j}^S$:

$$\begin{aligned} T_{ij} \;\; = \;\; \Big\{ & \varphi \mid S(\varphi) = i, \ \exists H/C \in D \text{ and } \sigma : V \longrightarrow C(\Sigma) \text{ with } \sigma(C) = \varphi, \\ & \forall\, h \in \mathrm{PF}(H) : U_{i,j} \cup U_i \vdash \sigma(h) \text{ and } \forall\, h \in \mathrm{NF}(H) : U_i \vdash \sigma(h) \Big\}. \end{aligned}$$

The next theorem is taken from Verhoef [8] but its proof is essentially the same as a similar theorem of Groote [10].

Theorem 2.11 *Let $T = (\Sigma, D)$ be a term deduction system and let $S : \mathrm{PF}(T) \longrightarrow \alpha$ be a stratification for an ordinal number α. Then T_S agrees with T. If S' is also a stratification for T then $T_S = T_{S'}$.*

Example 2.12 *Since our running example is stratifiable it follows from the above theorem that the rules in table 1 determine a transition relation (with predicates) on closed terms.*

Definition 2.13 *Let $T = (\Sigma, D)$ be a term deduction system with $D = D(T_p, T_r)$. Let in the following K, L, M, and N be index sets of arbitrary cardinality, let $s_k, t_l, u_m, v_n, t \in O(\Sigma)$ for all $k \in K$, $l \in L$, $m \in M$, and $n \in N$, let $P_k, P_m, P \in T_p$ be predicate symbols for all $k \in K$ and $m \in M$, and let $R_l, R_n, R \in T_r$ be relation symbols for all $l \in L$ and $n \in N$.*

A deduction rule $d \in D$ is in pntyft[1] *format if it has the form*

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{C}$$

with $C = f(x_1, \ldots, x_n)Rt$, $f \in \Sigma$ an n-ary function symbol and $X \cup Y = \{x_1, \ldots, x_n\} \cup \{y_i : i \in I\} \subseteq V$ a set of distinct variables. If $var(d) = X \cup Y$ we call d pure. *A variable in $var(d)$ that does not occur in $X \cup Y$ is called* free.

A deduction rule $d \in D$ is in pntyxt *format if it has the form above and $C = xRt$. $X \cup Y = \{x\} \cup \{y_i : i \in I\} \subseteq V$ is a set of distinct variables. If $var(d) = X \cup Y$ we call d pure. A variable in $var(d)$ that does not occur in $X \cup Y$ is called free.*

A deduction rule is in pntyf *format if it has the form above and $C = Pf(x_1, \ldots, x_n)$ and it is in* pntyx *format if it has the form above and $C = Px$. with $X \cup Y \subseteq V$ distinct variables. The notions pure and free are defined as expected.*

If a deduction rule $d \in D$ has one of the above forms we say that this rule is in panth *format, which stands for "predicates and* ntyft/ntyxt *hybrid format". A term deduction system is in* panth *format if all its rules are. A term deduction system is called pure if all its rules are pure. The* ntyft/ntyxt *format is defined in Groote [10].*

Example 2.14 *The operational rules of our running example satisfy the* panth *format and are pure.*

---

[1] Don't worry, there's no need to remember such double Dutch.

Definition 2.15 *Let $T = (\Sigma, D)$ be a term deduction system and let F be a set of formulas. The* variable dependency graph *of F is a directed graph with variables occurring in F as its nodes. The edge $x \longrightarrow y$ is an edge of the variable dependency graph if and only if there is a positive relation $tRs \in F$ with $x \in var(t)$ and $y \in var(s)$.*

The set F is called well-founded *if any backward chain of edges in its variable dependency graph is finite. A deduction rule is called well-founded if its set of hypotheses is so. A term deduction system is called well-founded if all its deduction rules are well-founded.*

Example 2.16 *It is not hard to see that the rules of our running example are well-founded.*

3. OPERATIONAL AND EQUATIONAL CONSERVATIVITY

In this section we prove a general operational conservative extension theorem with easy to check conditions. As a corollary, we prove a general equational conservative extension theorem. If we moreover have the elimination property for the new operators we also have completeness of the extension. We use the example of the previous section to elucidate the definitions. Moreover, we demonstrate the two theorems by applying them on the running example.

Definition 3.1 *Let Σ_0 and Σ_1 be signatures. If for all $f \in \Sigma_0 \cap \Sigma_1$ the arity of f in Σ_0 is the same as the arity of f in Σ_1 then $\Sigma_0 \oplus \Sigma_1$, called the* sum *of Σ_0 and Σ_1, is the signature $\Sigma_0 \cup \Sigma_1$. Note that $\oplus$ is not simply the union of two signatures: if Σ_0 contains a unary f and Σ_1 contains a binary f, their sum is* not *defined (but their union is).*

Example 3.2 *Let $\Sigma_0 = \{a : a \in A\} \cup \{+, \cdot\}$ and $\Sigma_1 = \{\sigma_d, +, \cdot\}$ be signatures. Then $\Sigma_0 \oplus \Sigma_1$ is defined and equals the signature of our running example. Moreover, Σ_0 is the signature of untimed subtheory, which is known as* BPA*.*

Definition 3.3 *Let $T^i = (\Sigma_i, D_i)$ be term deduction systems with predicate and relation symbols T^i_p and T^i_r respectively ($i = 0, 1$). Let $\Sigma_0 \oplus \Sigma_1$ be defined. The sum $T^0 \oplus T^1$, called the sum of T^0 and T^1, is the term deduction system $(\Sigma_0 \oplus \Sigma_1, D_0 \cup D_1)$ with predicate and relation symbols $T^0_p \cup T^1_p$ and $T^0_r \cup T^1_r$.*

Example 3.4 *Let T_0 be the term deduction system with Σ_0 of 3.2 and the rules without a $\xrightarrow{\sigma}$ of table 1 and let T_1 have Σ_1 of 3.2 as its signature and the rules with $\xrightarrow{\sigma}$ of table 1. Then the sum of T_0 and T_1 is defined. The sum is the operational semantics of our running example and T_0 is the operational semantics of the untimed subtheory:* BPA*.*

3.1. Operational conservativity

Next, we formally define the notion of an operational conservative extension and the notion of an operational conservative extension up to some semantical equivalence which is defined exclusively in terms of predicate and relation symbols. The notions operational conservative extension and operational conservative extension up to strong bisimulation equivalence were already defined by Groote and Vaandrager [1] (without the adjective 'operational') where they used the first notion to characterize the completed trace congruence induced by their pure well-founded *tyft/tyxt* format. Groote [10] gives the two definitions in the case that negative premises come into play. He used operational conservativity for a similar characterization result as in [1]. In Bol and Groote [11] the approach of Groote [10] is placed in a wider perspective. Aceto, Bloom and Vaandrager [12] use a restricted form of operational conservative extension for technical reasons; they call it disjoint extension. We will use the notion of operational conservativity to prove equational conservativity.

Definition 3.5 *Let $T^i = (\Sigma_i, D_i)$ be term deduction systems with $T = (\Sigma, D) := T^0 \oplus T^1$ defined. Let $D = D(T_p, T_r)$. The term deduction system T is called an* operational conservative extension *of T^0 if it is stratifiable and for all $s, u \in C(\Sigma_0)$, for all relation symbols $R \in T_r$ and predicate symbols $P \in T_p$, and for all $t \in C(\Sigma)$ we have*

$$T_S \vdash sRt \iff T^0_{S^0} \vdash sRt \text{ and } T_S \vdash Pu \iff T^0_{S^0} \vdash Pu$$

where S is a stratification for T and S^0 is a stratification for T^0 (take for instance S^0 to be the restriction of S to positive formulas of T^0).

Definition 3.6 *Let $T^i = (\Sigma_i, D_i)$ be term deduction systems with $T = (\Sigma, D) := T^0 \oplus T^1$ defined. If we have for all $s, t \in C(\Sigma_0)$ that $s =^{\oplus}_{\varphi} t \iff s =^{0}_{\varphi} t$ we say that T is an* operational conservative extension of T_0 up to φ equivalence*; where φ is some semantical equivalence that is defined in terms of relation and predicate symbols only.*

Remark 3.7 *Many equivalences are definable in terms of relation and predicate symbols only: for instance, trace equivalence, completed trace equivalence, failure equivalence, readiness equivalence, failure trace equivalence, ready trace equivalence, possible-future equivalence, simulation equivalence, complete simulation equivalence, ready simulation equivalence, nested simulation equivalence, strong bisimulation equivalence, weak bisimulation equivalence, η bisimulation equivalence, delay bisimulation equivalence, branching bisimulation equivalence, the rooted variants of the last four equivalences, and more equivalences. We refer to Van Glabbeek's linear time – branching time spectra [6] and [7] for more information on these equivalences. In the applications we also make use of ε bisimulation originating from Koymans and Vrancken [17]. For all these equivalences, we have that the following theorem holds. It states that if an extension is operationally conservative, it is also operationally conservative up to some equivalence definable in terms of relations and predicates only.*

Theorem 3.8 *Let $T^i = (\Sigma_i, D_i)$ be term deduction systems and let $T = T^0 \oplus T^1$ be defined. If T is an operational conservative extension of T_0 then it is also an operational conservative extension up to φ equivalence; where φ is an equivalence relation defined exclusively in terms of predicate and relation symbols.*

Proof. (Sketch). Let $s, t \in C(\Sigma_0)$ be original closed terms. Since T is an operational conservative extension of T_0 we have that the process graphs (or, better, term-relation-predicate diagrams; defined in the obvious way) of s in both T and T_0 are the same. So if we partition the set of these graphs with some equivalence defined exclusively in terms of relations and predicates then s and t are in the same part in the extended system T if and only if they are in the same part in the original system T_0. ∎

The next theorem gives sufficient conditions such that $T_0 \oplus T_1$ is an operational conservative extension of T_0. The theorem is a threefold generalization of a similar result for Groote's *ntyft/ntyxt* format [10]. Firstly, because we prove it for the *panth* format, which is a generalization of the *ntyft/ntyxt* format. Secondly, since we allow new rules to contain original function symbols in the left-hand side of a conclusion such as, for instance, the last four rules in table 1 of our running example. This is not allowed in the setting of Groote. Thirdly, since Groote requires for the new rules that the left-hand side of a conclusion may not be a single variable, whereas we do not have such a restriction. The first such theorem was formulated in Groote and Vaandrager [1].

Theorem 3.9 *Let $T^0 = (\Sigma_0, D_0)$ be a pure well-founded term deduction system in* panth *format. Let $T^1 = (\Sigma_1, D_1)$ be a term deduction system in* panth *format. If there is a conclusion sRt or Ps of a rule $d_1 \in D_1$ with $s = x$ or $s = f(x_1, \ldots, x_n)$ for an $f \in \Sigma_0$, we additionally require that d_1 is pure, well-founded, $t \in O(\Sigma_0)$ for premises tRy of d_1, and that there is a positive premise containing only Σ_0 terms and a new relation or predicate symbol. Now if $T = T^0 \oplus T^1$ is defined and stratifiable then T is an operational conservative extension of T^0.*

Proof. Let $T = (\Sigma, D)$ and $D = D(T_p, T_r)$. Let $S : PF(T) \longrightarrow \alpha$ be a stratification for T and let $S^0 : PF(T^0) \longrightarrow \alpha$ be the restriction of S to $PF(T^0)$ (note that S^0 is a stratification).

Let $u, w \in C(\Sigma_0)$, $R \in T_r$, $P \in T_p$, and $v \in C(\Sigma)$. We are to show that the following two bi-implications hold

$$\begin{aligned} T_S \vdash uRv &\iff T^0_{S^0} \vdash uRv, \\ T_S \vdash Pw &\iff T^0_{S^0} \vdash Pw. \end{aligned}$$

By definition 2.10 it suffices to prove the two bi-implications below for all $i < \alpha$.

$$T_i \vdash uRv \iff T^0_i \vdash uRv, \tag{1}$$
$$T_i \vdash Pw \iff T^0_i \vdash Pw. \tag{2}$$

We will do this by transfinite induction on i. So let both statements be true for all $i' < i$ then we prove them for i.

We begin to prove both implications from left to right. By definition 2.10 it suffices to show for all $j < d(T)$ that

$$T_{i,j} \vdash uRv \implies T^0_i \vdash uRv, \tag{3}$$
$$T_{i,j} \vdash Pw \implies T^0_i \vdash Pw. \tag{4}$$

We will do this by transfinite induction on j. So let 3 and 4 be true for all $j' < j$. We prove them for j. By definition 2.10 there is a rule $d \in D$

$$\frac{\{P_k s_k : k \in K\} \cup \{t_l R_l y_l : l \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n \neg R_n : n \in N\}}{C} \tag{5}$$

with $C = sRt$. There is also a closed substitution σ with $\sigma(s) = u$ and $\sigma(t) = v$. We first show that $d \in D_0$. Suppose that this is not the case. Since $u \in C(\Sigma_0)$ we must have that $s = x$ or that $s = f(x_1, \ldots, x_n)$ for some $f \in \Sigma_0$; so the additional requirements clearly hold for d. Let $var(s) = X$ and $Y = \{y_l : l \in L\}$. Since d is pure we have that $var(d) = X \cup Y$. We know that $\sigma(x) \in C(\Sigma_0)$ for all $x \in X$. We show that for all $y \in Y$ we have $\sigma(y) \in C(\Sigma_0)$. Suppose that there is a $y_{l_0} \in Y$ with $\sigma(y_{l_0}) \in C(\Sigma) \setminus C(\Sigma_0)$. This contradicts the well-foundedness of the rule d, for $U_i \cup U_{i,j} \vdash \sigma(t_{l_0}) R_{l_0} \sigma(y_{l_0})$ so with the induction hypotheses on i or j we find that $\sigma(t_{l_0}) \in C(\Sigma) \setminus C(\Sigma_0)$. Since t_{l_0} is a Σ_0 term, this must be the result of a substitution. This can only be due to a variable $y_{l_1} \in Y$. With induction on the subsubscript we find an infinite backward chain of edges $y_{l_0} \longleftarrow y_{l_1} \longleftarrow \ldots$ in the variable dependency graph of d. So $\sigma(y) \in C(\Sigma_0)$ for all $y \in Y$. Now let h be a positive premise containing only Σ_0 terms and a new relation or predicate symbol. By definition 2.10 we have $U_i \cup U_{i,j} \vdash \sigma(h)$ so with induction on i or j we find that $U_i^0 \cup U_{i,j}^0 \vdash \sigma(h)$, which is a contradiction since the $\sigma(h)$ is not even a formula in T^0. So the assumption that $d \in D_1$ cannot hold and we must have that $d \in D_0$.

This means that d is pure and well-founded. Just as above we can show that $\sigma(x) \in C(\Sigma_0)$ for all $x \in X \cup Y$ so we have that all the instantiated premises of d only contain Σ_0 terms. So we find with induction on i and/or j that for all positive premises h of rule d we have $U_i^0 \cup U_{i,j}^0 \vdash \sigma(h)$. Suppose that $U_i^0 \nvdash \sigma(v_n \neg R_n)$. Then there is a $v_n' \in C(\Sigma_0)$ such that $U_i^0 \vdash \sigma(v_n R_n v_n')$ so with induction on i we find that also $U_i \vdash \sigma(v_n R_n v_n')$, which is a contradiction. In this way we find that $U_i^0 \vdash \sigma(h)$ for all negative premises h of rule d. By definition 2.10 we have $T_{i,j}^0 \vdash uRv$ so $T_i^0 \vdash uRv$.

The case $C = Ps$ is treated in the same way. This ends our induction step on j, which proves 3 and 4. So we find that equations 1 and 2 hold from left to right for i.

Now we show that they hold from right to left for i. By definition 2.10 it suffices to show for all $j < d(T^0)$ that

$$\begin{aligned} T_{i,j}^0 \vdash uRv &\implies T_i \vdash uRv, \\ T_{i,j}^0 \vdash Pw &\implies T_i \vdash Pw. \end{aligned}$$

This can be proved with induction on j in the same way as we proved both implications from left to right, but simpler since we can apply induction immediately. This concludes the proof. ∎

Example 3.10 *It is not hard to see that the term deduction systems T_0 and T_1 of example 3.4 satisfy all the conditions of theorem 3.9. This means that the term deduction system belonging to BPA_{dt} is an operational conservative extension of the one that belongs to* BPA. *Now with theorem 3.8 we find that BPA_{dt} is an operational conservative extension up to strong bisimulation equivalence.*

3.2. Equational conservativity

We devote the remainder of this section to equational conservativity. We recall that an equational specification is a pair consisting of a signature and a set of equations over this signature.

Definition 3.11 *Let $L_i = (\Sigma_i, E_i)$ be equational specifications ($i = 0, 1$). Let $\Sigma_0 \oplus \Sigma_1$ be defined. Then the sum $L_0 \oplus L_1$ of L_0 and L_1 is the equational specification $(\Sigma_0 \oplus \Sigma_1, E_0 \cup E_1)$.*

Table 2: The axioms of BPA and DT.

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$\sigma_d(x) + \sigma_d(y) = \sigma_d(x + y)$	DT1
$\sigma_d(x) \cdot y = \sigma_d(x \cdot y)$	DT2

Example 3.12 *Let Σ_0 and Σ_1 be the signatures of 3.2. Let E_0 consist of the axioms A1–A5 in table 2 and let E_1 contain the two discrete time axioms DT1 and DT2 in that same table. Then (Σ_0, E_0) is the equational specification* BPA. *We define DT to be the equational specification (Σ_1, E_1). The equational specification BPA_{dt} of our running example is just* BPA $\oplus$ *DT.*

Next, we recall the well-known definition of conservativity.

Definition 3.13 *Let $L_i = (\Sigma_i, E_i)$ be equational specifications and let $L = L_0 \oplus L_1$ be defined. We say that L is an* equational conservative extension*, or simply a* conservative extension*, if for all $s, t \in C(\Sigma_0)$*

$$L \vdash s = t \iff L_0 \vdash s = t.$$

If for all $s \in C(\Sigma)$ there is a $t \in C(\Sigma_0)$ such that $L \vdash s = t$ we say that L has the elimination property*.*

Theorem 3.14 *Let $L_i = (\Sigma_i, E_i)$ be equational specifications and let $L = (\Sigma, E) = L_0 \oplus L_1$ be defined. Let $T_i = (\Sigma_i, D_i)$ be term deduction systems and let $T = T_0 \oplus T_1$. Let φ be an equivalence that is definable in terms of predicate and relation symbols only. Let E_0 be a complete axiomatization with respect to the φ equivalence model induced by T_0 and let E be a sound axiomatization with respect to the φ equivalence model induced by T. If T is an operational conservative extension of T_0 up to φ equivalence then L is an equational conservative extension of L_0.*

Moreover, if L has the elimination property we have that E is a complete axiomatization with respect to the φ equivalence model induced by T.

Proof. The implication from right to left is trivial, so we only treat the other case. Let $s, t \in C(\Sigma_0)$ and suppose $L \vdash s = t$. Since E is sound we have that $s =^{\oplus}_{\varphi} t$. Since T is an operational conservative extension of T_0 up to φ equivalence we have that $s =^{0}_{\varphi} t$. Since E_0 is complete with respect to φ equivalence we now have that $L_0 \vdash s = t$.

Now suppose moreover that for all $s \in C(\Sigma)$ there is a $t \in C(\Sigma_0)$ such that $L \vdash s = t$. We show that E is complete. Let s and t be φ equivalent Σ terms. Then there are $s', t' \in C(\Sigma_0)$ such that $L \vdash s = s'$ and $L \vdash t = t'$. With the soundness of E we find that s' and t' are φ equivalent in T and since this is an operational conservative extension of T_0

up to φ equivalence, we find that s' and t' are φ equivalent in T_0. Since E_0 is complete, we find $L_0 \vdash s' = t'$ so we find $L \vdash s = t$. ∎

Example 3.15 *It is well-known that the axioms of* BPA *are a complete axiomatization with respect to strong bisimulation semantics; see for instance [18]. Baeten and Bergstra [15] show that BPA_{dt} is sound with respect to bisimulation semantics. So since the term deduction system belonging to BPA_{dt} is an operational conservative extension of the term deduction system of* BPA *(see example 3.10) we may conclude with theorem 3.8 and theorem 3.14 that $BPA_{\mathrm{dt}} = \mathrm{BPA} \oplus DT$ is an equational conservative extension of* BPA. *Incidentally, this result is new.*

4. GROUND CONFLUENCE MODULO EQUATIONS

As a first application of our conservativity results we prove a general reduction theorem stating that in many cases checking the Church-Rosser property for closed terms modulo some equations for a large system reduces to verifying this property for a small basic system. Of course, provided that the large system is an equational conservative extension of the small system. From a term rewriting point of view this condition is not realistic since usually you need the Church-Rosser property for closed terms to obtain conservativity. However, in many cases we can prove the conservativity without a term rewriting analysis. Thus, we could argue that conservativity and ground confluence are equally powerful properties, so to speak.

Definition 4.1 *A* term rewriting system *is a pair (Σ, R) with Σ a signature and R a set of* rewrite rules. *Rewrite rules are pairs of terms (over Σ) that we denote $s \to t$. We suppose that s is not a variable and that $var(t) \subseteq var(s)$. The* one step rewrite relation $\to^1_R$ *is the smallest relation on terms containing R that is closed under substitutions and contexts. The* rewrite relation $\to_R$ *is the transitive-reflexive closure of the one step rewrite relation $\to^1_R$. Often, we refer to a term rewriting system (Σ, R) by its set of (rewrite) rules R.*

Definition 4.2 *Let R be a set of rewrite rules and E be a set of equations. Let $=_E$ be the smallest congruence generated by the equations in E. The one step rewriting relation $\to^1_{R/E}$ is defined as $=_E \circ \to^1_R \circ =_E$. The rewriting relation $\to_{R/E}$ is the transitive-reflexive closure of the one step rewrite relation $\to^1_{R/E}$. We recall that for two relations R and S we have $R \circ S = \{(r, s) \mid \exists t : (r, t) \in R, (t, s) \in S\}$.*

Definition 4.3 *Let R be a set of rules and let E be a set of equations. Let $\to$ be the rewriting relation $\to_{R/E}$. Let s be a term. If for all s_0, s_1 such that $s \to s_0$ and $s \to s_1$ there is a term s' such that $s_0 \to s'$ and $s_1 \to s'$ we say that the rewriting relation $\to_{R/E}$ is* Church-Rosser *or* confluent. *We call $\to_{R/E}$* ground Church-Rosser *if it is Church-Rosser for closed terms. Sometimes, we will write* CR *instead of Church-Rosser. We also say that $\to_R$ is Church-Rosser (or confluent) modulo E; we write* CR/E. *In the literature we also see E-Church-Rosser and E-confluence if $\to_{R/E}$ is confluent in the above sense; see, for instance, Jouannaud and Muñoz [19].*

Definition 4.4 *Let R be a set of rules and let E be a set of equations. Let $=$ be the least congruence generated by the equations in E and the rules in R in both ways. We say that the rewriting relation $\to_{R/E}$ is ground* $\mathrm{CR}^=$ *if for all ground terms s and t such that $s = t$ there are terms s' and t' such that $s \twoheadrightarrow_{R/E} s'$, $t \twoheadrightarrow_{R/E} t'$, and $s' =_E t'$.*

Remark 4.5 *It is easily seen that $\to_R$ is ground* CR/E *if and only if it is ground* $\mathrm{CR}^=$/E.

Definition 4.6 *A term rewriting system R is* terminating *if there exists no infinite sequence $s_0 \to_R^1 s_1 \to_R^1 s_2 \ldots$. We call a term s a normal form if we do not have $s \to_R^1 s'$ for any s'.*

Theorem 4.7 *Let $L_i = (\Sigma_i, E_i)$ be equational specifications and let $L = (\Sigma, E) = L_0 \oplus L_1$ be defined. Suppose that L is a conservative extension of L_0. Turn a set $R_0^= \subseteq E_0$ into a set of rewrite rules R_0 and let $A_0 = E_0 \setminus R_0^=$ be a set of equations (or axioms). Turn the set $R^= = (E \setminus A_0) \cup R_0^=$ into a set of rewrite rules R. Suppose that $\to_R$ is terminating and that normal forms are Σ_0 terms (so L has the elimination property). If $\to_{R_0/A_0}$ is ground Church-Rosser then $\to_{R/A_0}$ is also ground Church-Rosser.*

Proof. Let s and t be ground Σ terms and suppose that $E \vdash s = t$. By assumption, there are ground Σ_0 terms s' and t' with $s \twoheadrightarrow_R s'$ and $t \twoheadrightarrow_R t'$. So $E \vdash s' = t'$. Since L is a conservative extension of L_0 we now have that $E_0 \vdash s' = t'$. Since $\to_{R_0/A_0}$ is ground CR there are s_0 and t_0 such that $s' \twoheadrightarrow_{R_0/A_0} s_0$, $t' \twoheadrightarrow_{R_0/A_0} t_0$, and $A_0 \vdash s_0 = t_0$. Since $R_0 \subseteq R$ we also have $s' \twoheadrightarrow_{R/A_0} s_0$. Since $s \twoheadrightarrow_R s'$ we also have $s \twoheadrightarrow_{R/A_0} s'$ (simply put $s =_{A_0} s, \ldots, s' =_{A_0} s'$ between the one step rewritings). So we find that $s \twoheadrightarrow_{R/A_0} s_0$. In the same way we find that $t \twoheadrightarrow_{R/A_0} t_0$, and we have $A_0 \vdash s_0 = t_0$. This implies using remark 4.5 that $\to_{R/A_0}$ is ground CR. ∎

5. APPLICATIONS

In this section we will give the reader an idea of the applicability of our conservativity results, the completeness corollary and the ground Church-Rosser reduction theorem. Noteworthy perhaps, is that we could not find any equational conservativity result in the literature for which our method does not work, as well. We apply our results to various real and discrete time settings in CCS, ACP, and ATP. We discuss applications of our theorems involving the notions projection, renaming, state operator, priority, recursion, the silent step (both the weak and branching variants), the empty process, and combinations of these notions in the setting of ACP languages. And we mention applications concerning the delay operators of ATP. Before we continue, we recall that we already discussed an application in great detail, namely our running example $\mathrm{BPA_{dt}}$.

5.1. The operational approach

Moller and Tofts [20] discuss an extension of CCS with time: TCCS. Their approach is an operational one, that is, they add operational rules to the well-known operational rules of CCS to define new operators and to extend the meaning of existing operators. With our operational conservativity result it is easily seen that TCCS is an operational conservative extension of CCS. An interesting matter is that Moller and Tofts take for their equational approach a small sublanguage of TCCS, which is a mixture of timed and untimed operators. In fact, they took a small one which is sound and complete with respect to the operational rules such that extensions of this sublanguage have the elimination property, thereby reducing the completeness to the sublanguage. They use a variant of strong bisimulation equivalence, which is definable in terms of transition relations (and predicates) only. So, their extensions are easily seen to be conservative with our theorems. Our operational conservative extension theorem is a useful tool to systematically find such small sublanguages such that their extensions are conservative.

5.2. The axiomatical approach

The approach in ACP and ATP is more axiomatical. This is no obstacle for our results: we can also handle this approach in a satisfactory way. Within the ACP community there is a long tradition with conservativity results, completeness results and confluence results. Also in ATP there are many conservativity and completeness results. We will simultaneously treat numerous examples from both ACP and ATP with the aid of figure 1. And we will treat some typical cases more elaborately. We note that the examples in figure 1 contain both known results and new results.

In the introduction we mentioned the problems concerning the confluence of ACP that Bergstra and Klop [2] used to prove conservativity. We claimed that with our theorems it is very easy to see that the conservativity result holds. Therefore, we elaborately treat the •-labelled arrow from ACP to BPA_δ in figure 1. We show that all our general results apply to this arrow.

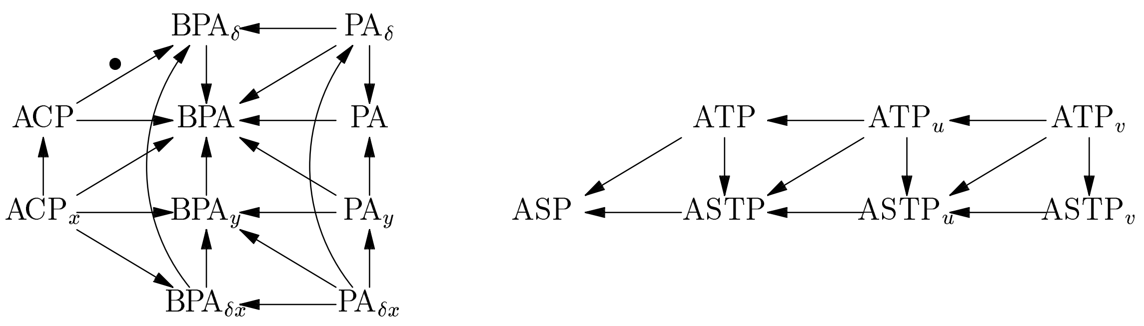
Van Glabbeek [21] gives an operational semantics for Bergstra and Klop's ACP [2] and for their sequential subsystem BPA_δ [2]. With our operational result 3.9 it is easily seen that the large semantics is an operational conservative extension of the small one. Baeten and Weijland [18], for instance, show that BPA_δ is sound and complete with respect to the small semantics and that ACP is sound with respect to the large one. They use a variant of strong bisimulation with successful termination predicates, which is definable in terms of transition relations and predicates only. So, our equational result 3.14 immediately implies that ACP is an equational conservative extension of BPA_δ. Since ACP has the elimination property we also find the completeness of ACP with theorem 3.14. Moreover, with our reduction theorem 4.7 we have that the question whether or not ACP is ground Church-Rosser modulo associativity and commutativity of the choice (CR/AC) reduces to this question for BPA_δ. The associated term rewriting system of BPA_δ consists of five rewrite rules and two equations, which is a considerable reduction since the term rewriting system for ACP has many more rules.

Now, we discuss figure 1. An arrow $A \longrightarrow B$ indicates that system A is both an operational and an equational conservative extension of system B and that this can be shown using our conservativity results. We will explain the abbreviations that we did not met yet. The abbreviation PA stands for process algebra and this system originates from Bergstra and Klop [22]. The acronym ASP stands for the algebra of sequential processes. This system stems from Milner [23]. The x, y, u and v stand for variables; we use them to treat many examples at the same time.

We begin with the variables x and y; they are present in the ACP-side of figure 1.

Let $x = y$. And let x be one of PR, RN, λ, Λ, θ, or a combination of them. The abbreviations stand for projections, renamings, simple state operators, extended state

Figure 1: Some applications.



operators, and the priority operator respectively. A concise reference to these notions, their operational rules, their axiomatizations, and their associated term rewriting systems is the text book of Baeten and Weijland [18]. The variant of bisimulation that is used in these applications is definable in terms of transition relations and predicates exclusively. So, for all these cases we have that all arrows of the ACP side of the picture hold: operational and equational conservativity. Moreover, all these extensions have the elimination property for either the complete BPA or the complete BPA_δ (if the extension contains already a δ); for full proofs see, for instance, [18]. So we find for all these extensions the completeness with our corollary. Moreover, the ground confluence modulo AC for these systems reduces to the ground confluence modulo AC for either BPA or BPA_δ.

Now, let $x = y$ and let x be one of rec, dt, or a combination of those (note that we can also combine rec with the already treated notions). The abbreviations stand for recursion and discrete time [15], respectively. Also for these systems we have that all arrows hold. Note that $\mathrm{BPA}_{\mathrm{dt}} \longrightarrow \mathrm{BPA}$ was our running example. We do not have the elimination property for subscripted systems to systems without a subscript (for instance $\sigma_d(a)$ cannot be written as a BPA term). For the other arrows we have the elimination property [15], so from the completeness of BPA_y we conclude the completeness for all the extensions. The ground confluence of these systems has not yet been studied but with our reduction theorem is it only necessary to study the ground confluence for the BPA_y systems.

Now, let $x = y$ and let x be Milner's silent action τ. We already mentioned in the introduction that systems containing the three τ laws of Milner have in general bad rewriting properties. The conservativity of ACP_τ over ACP was proved semantically by Bergstra and Klop [4] since the second and third τ law have no clear term rewriting direction. Next, we will show that our approach also works in cases where the established method breaks down. In fact, we immediately find this result. The operational semantics of ACP_τ is just the one of ACP but now a ranges also over τ itself. It is easy to see that the conditions of theorem 3.9 are satisfied, so ACP_τ is an operational conservative extension of ACP. Now with theorem 3.8 we find that ACP_τ is an operational conservative extension up to rooted τ bisimulation equivalence of ACP. We should note that the definition of (rooted) τ bisimulation equivalence in the presence of the predicates $\xrightarrow{a}\surd$ as given by Baeten and Weijland [18] is wrong, since the first τ law of Milner is not sound. This can be easily repaired. Since ACP is sound and complete and since ACP_τ is sound with respect to this equivalence, we find with theorem 3.14 that ACP_τ is an equational conservative extension of ACP. All the other arrows in our figure go likewise. Since all the extensions have the elimination property for $\mathrm{BPA}_{\delta,\tau}$, we find their completeness with the aid of the completeness of $\mathrm{BPA}_{\delta,\tau}$. The systems have bad term rewriting properties so the ground confluence results does not apply.

We mentioned in the introduction the rather complicated term rewriting analysis of Akkerman and Baeten [3] of a fragment of ACP with the branching τ. We will show in a moment that our results can be easily applied to this case. With the aid of theorem 3.8 we find that ACP_τ is an operational conservative extension up to branching bisimulation equivalence [24] of ACP. Also in this case we note that the definition of branching bisimulation equivalence in the presence of the predicates $\xrightarrow{a}\surd$ as given by Baeten and Weijland [18] is wrong, since the first τ law of Milner is not sound. Having repaired this definition, we find in the same way as above that ACP with the branching τ axioms [24], denoted ACP^τ, is an equational conservative extension of ACP. The same holds for all the other arrows in our figure. Since all the extensions have the elimination property for BPA^τ we find the completeness for them with the completeness of BPA^τ. The branching τ axioms have better term rewriting properties [3] than the τ laws of Milner (that we discussed above). So our ground confluence result may be useful, as well.

Let $x = y$ be the empty process ε of Koymans and Vrancken [17]; see also Vrancken [25]. We can show operational and equational conservativity for all arrows from a system with an ε to a subsystem also featuring this ε by using the operational semantics that can be found in Baeten and Weijland's text book [18]. In [18] we also find that these systems have the elimination property, so also our completeness and the ground confluence results apply. For the remaining arrows we have to follow a different approach. The operational semantics in [18] features the rule $a \xrightarrow{a} \varepsilon$ so we can never have that this semantics is an operational conservative extension of a semantics without ε (but containing a). For, there is no ε in the subsystem. The solution to this problem is to take another operational semantics that is easily obtained by "upgrading" the complete graph model of Koymans and Vrancken [17]. In fact, this operational semantics is that of the subsystem where we include ε as a normal atomic action. So we have, for instance, $\varepsilon \xrightarrow{\varepsilon}\surd$. The special behaviour of the empty process is expressed with the aid of so-called ε bisimulation equivalence of Koymans and Vrancken [17]. Also this definition needs a straightforward upgrade from graphs to transitions (and is definable in terms of transition relations and predicates only). In this way we find the operational and equational conservativity. Since we cannot eliminate the empty process, we cannot apply our completeness corollary and the ground confluence result for these particular systems.

Let x be ρ standing for absolute real time [26]. Then the x-arrow in the figure holds. To obtain this result we take the operational semantics of Klusener [27]. Also here we have the elimination property, so our completeness and ground confluence results apply, too.

Now we treat the ATP-side of figure 1. Nicollin and Sifakis [9] studied a timed process algebra called ATP with various extensions and restrictions of which the most restricted timed one is ASTP, the algebra of sequential timed processes. Milner's [23] algebra of sequential processes ASP—the untimed version of ASTP—is the most restricted system. The interesting thing here is that they prove some conservativity results with the same strategy as ours: they show that the extensions are operationally conservative up to bisimulation by looking at the transition rules and then conclude the equational conservativity. Our figure intends to show that every possible extension that can be obtained with the so-called delay operators of Nicollin and Sifakis [9] is conservative. However, we should note that we treat recursion as in Groote and Vaandrager [1] to fit the operational semantics in their *tyft/tyxt* format—and thus Verhoef's *panth* format—in order to obtain that all our conditions are satisfied. This is not a serious drawback but just an equivalent treatment of recursion. There are four delay operators present in [9]: start delay, unbounded start delay, execution delay, and termination delay. The termination delay (td) is an enhancement of the execution delay (ed) so if we have the termination delay we also have execution delay. For u we can take any combination of delay operators. If u does not contain all delay operators yet we can take for v the operators of u and a new one, or if the execution delay operator is in u we can take the termination delay operator in v and we do not necessarily need an extra delay operator for a non-trivial extension. Cases like ASP$\longrightarrow$ASTP and $\mathrm{ASTP}_{\mathrm{td}}\longrightarrow\mathrm{ASTP}_{\mathrm{ed}}$ are, in our opinion, the most interesting since in these cases not only a new operator (unit delay and ξ resp.) is introduced but also an original operator gets a new rule. Since the elimination property holds [9] for ASTP our completeness corollary applies for all the arrows but the two to ASP. The ground confluence of ATP is not yet studied but its study reduces to that of ASTP with our reduction theorem.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented general conservativity results for transition system based process theories with reasonable and easy-to-check conditions. As a simple corollary of the conservativity results we proved a completeness theorem. As a first application we proved a general theorem giving sufficient conditions to reduce the question of ground confluence modulo some equations for a large term rewriting system associated with an equational process theory to a small term rewriting system under the condition that the large system is a conservative extension of the small one. With numerous examples that we took from the literature about CCS, ACP, and ATP we showed that our theorems are useful. The applications include various real and discrete time settings in CCS, ACP, and ATP; the notions projection, renaming, state operator, priority, recursion, the silent step (both the weak and branching variants), the empty process, and combinations of these notions. We want to stress that the established method for proving conservativity in ACP makes use of a rather complicated term rewriting analysis, whereas our method is very easily applicable. This is a great advantage of our approach in our opinion.

Remarkably, we could not find any conservativity results in the literature for which our equational conservativity theorem does not apply, too. So it may be an interesting idea to investigate whether or not the conditions of this theorem are necessary. We think that our conditions are only sufficient but not necessary.

Acknowledgements Thanks to Jos Baeten (Eindhoven University of Technology) for valuable comments and interesting discussions. Thanks to Gertjan Akkerman (Delft University of Technology), Jean-Pierre Jouannaud (LRI Paris), Jan Willem Klop (CWI), and Xavier Nicollin (IMAG France) for their helpful discussions.

REFERENCES

1. J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.
2. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.
3. G.J. Akkerman and J.C.M. Baeten. Term rewriting analysis in process algebra. Report P9006, Programming Research Group, University of Amsterdam, 1990.
4. J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.
5. G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
6. R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, 1990.
7. R.J. van Glabbeek. The linear time – branching time spectrum II (extended abstract). In *Proceedings CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, August 1993.
8. C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. Computing Science Notes 93/18, Eindhoven University of Technology, 1993. To appear in the proceedings of CONCUR 94.
9. X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application (revised version). Technical Report RT-C26, LGI-IMAG, Grenoble, France, November 1991.
10. J.F. Groote. Transition system specifications with negative premises. Report CS-R8950, CWI, Amsterdam, 1989. An extended abstract appeared in J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90*, Amsterdam, LNCS 458, pages 332–341. Springer-Verlag, 1990.
11. R.N. Bol and J.F. Groote. The meaning of negative premises in transition system specifications (extended abstract). In J. Leach Albert, B. Monien, and M. Rodríguez, editors, *Proceedings* 18^{th} *ICALP,* Madrid, volume 510 of *Lecture Notes in Computer Science*, pages 481–494. Springer-Verlag, 1991. Full version appeared as Report CS-R9054, CWI, Amsterdam, 1990.
12. L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. In *Proceedings* 7^{th} *Annual Symposium on Logic in Computer Science,* Santa Cruz, California, pages 113–124. IEEE Computer Society Press, 1992. Full version available as CWI Report CS-R9218, June 1992, Amsterdam. To appear in the LICS 92 Special Issue of *Information and Computation.*
13. D.J.B. Bosscher. Term rewriting properties of SOS axiomatizations. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 425–439. Springer-Verlag, 1994.
14. A.S. Klusener. *Models and axioms for a fragment of real time process algebra.* PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, December 1993.
15. J.C.M. Baeten and J.A. Bergstra. Discrete time process algebra. In W.R. Cleaveland,

editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 401–420. Springer-Verlag, 1992.

16. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: Preliminary report. In *Conference Record of the* 15^{th} *ACM Symposium on Principles of Programming Languages,* San Diego, California, pages 229–239, 1988. Full version available as Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990. Accepted to appear in *Journal of the ACM.*
17. C.P.J. Koymans and J.L.M. Vrancken. Extending process algebra with the empty process ε. Logic Group Preprint Series Nr. 1, CIF, State University of Utrecht, 1985. The State University of Utrecht is now called Utrecht University.
18. J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.
19. J.-P. Jouannaud and M. Muñoz. Termination of a set of rules modulo a set of equations. In R. E. Shostak, editor, *7th International Conference on Automated Deduction*, volume 170 of *Lecture Notes in Computer Science*, pages 175–193. Springer-Verlag, 1984.
20. F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 1990.
21. R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings STACS 87*, volume 247 of *Lecture Notes in Computer Science*, pages 336–347. Springer-Verlag, 1987.
22. J.A. Bergstra and J.W. Klop. Fixed point semantics in process algebras. Report IW 206, Mathematisch Centrum, Amsterdam, 1982.
23. R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
24. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.
25. J.L.M. Vrancken. *Studies in Process Algebra, Algebraic Specifications and Parallelism.* PhD thesis, University of Amsterdam, April 1991.
26. J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Journal of Formal Aspects of Computing Science*, 3(2):142–188, 1991.
27. A.S. Klusener. Completeness in real time process algebra. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 376–392. Springer-Verlag, 1991.