A general conservative extension theorem in process algebra
Verhoef, C.

Published: 01/01/1993

Citation for published version (APA):
A general conservative extension theorem in process algebra

by

C. Verhoef

93/38
This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors: prof.dr.M.Rem
        prof.dr.K.M.van Hee.
A general conservative extension theorem in process algebra

C. VERHOEF

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, NL-5500 MB Eindhoven, The Netherlands
e-mail: chris@win.tue.nl, fax: +31 40 463992

ABSTRACT. We proved 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 outgoing 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 proved a completeness theorem. As a first application of our conservativity results, 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. We provided 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), the empty process, (or combinations of these notions). Collation: pp. 17, ill. 2, tab. 2, ref. 29.

Key Words & Phrases: structured operational semantics, term deduction system, predicate, negative premise, stratification, operational conservativity, equational conservativity, completeness, ground Church-Rosser.

1980 Mathematics Subject Classification (1985 Revision): 68Q05, 68Q55.
CR Categories: D.3.1, F.1.1, F.3.2, F.4.3.

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

1. Introduction

Since the past few years people working in the area of process algebra have started to extending 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 outgoing 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. [19]). 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 framework 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 [10] 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 [20] 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 \( \tau \) is both terminating and confluent modulo associativity and commutativity of the alternative composition. Akkerman [2] 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 [4] 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 transpose the question of equational conservativity to that of operational conservativity rather than to perform a term rewriting analysis. The only thing that remains 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 [8] for a semantical term rewriting analysis). However, since the paper of Plotkin [27], 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, \( \eta \) bisimulation equivalence, delay bisimulation equivalence, branching bisimulation equivalence, and more equivalences. We refer to Van Glabbeek's linear time – branching time spectra [15] and [16] for more information on these equivalences. In [15] and [16], 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 [10] reduce to a term rewriting analysis with only five rewrite rules and two
A general conservative extension theorem . . . : 2. Some general SOS definitions

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 $\tau$ laws of Milner, where the term rewriting approach breaks down; see, e.g., [8]. We will treat these examples in the applications.

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 [28]. 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.

Related work

In this subsection we briefly mention related work. Nicollin and Sifakis [26] 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 [19] under the name conservativity. In Groote [18] and in Bol and Groote [12] this notion also appears. In all these papers this notion is used for a different purpose than ours. Aceto, Bloom and Vaandrager [1] 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 have 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 [28] since at the moment his 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 $\Sigma$ 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. We restrict ourselves to signatures that contain at least one constant symbol. The notion of a term (over $\Sigma$) 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 $O(\Sigma)$ (note that a closed term is also open). We also want to speak about variables occurring in terms: let $t \in O(\Sigma)$ then $\text{var}(t) \subseteq V$ is the set of variables occurring in $t$.

A substitution $\sigma$ 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 $\sigma$-image.
Definition (2.1)

A term deduction system is a structure \((\Sigma, D)\) with \(\Sigma\) 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 respectively the set of predicate symbols and the set of relation symbols. Let \(s, t, u \in O(\Sigma)\), \(P \in T_p\), 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 \(PF(S)\) for the subset of positive formulas of \(S\) and \(NF(S)\) for the subset of negative formulas of \(S\).

A deduction rule \(d \in D\) has the form

\[
H \quad \overline{\quad} \quad C
\]

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.

Example (2.2)

As a running example we present an operational semantics that originates from Baeten and Bergstra [5] of a basic process language with relative discrete time: BPA 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; parameter of the theory), alternative and sequential composition, and a unary operator \(\sigma_d\), 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 \(-\rightarrow\) for all \(a \in A\), a relation \(-\rightarrow^*\) with \(s \notin A\) and predicates \(-\rightarrow^*/\) for all \(a \in A\). The intended interpretation of \(x \rightarrow a\) is that a process \(x\) evolves into \(x'\) when executing atomic action \(a\). The meaning of \(x \rightarrow a\) is that process \(x\) terminates successfully after the execution of \(a\). With \(x \rightarrow a\) we mean that \(x\) evolves into \(x'\) by moving to the next time slice.

<table>
<thead>
<tr>
<th>(a \rightarrow^\Delta x')</th>
<th>(x \rightarrow^\Delta x')</th>
<th>(x \rightarrow^\Delta y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
<td>(z \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(z \rightarrow a)</td>
<td>(z \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
<tr>
<td>(x \rightarrow a)</td>
<td>(x \rightarrow a)</td>
<td>(y \rightarrow a)</td>
</tr>
</tbody>
</table>

Table 1. BPA with discrete time.

Definition (2.3)

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 \(PF(T)\) and the negative formulas by \(NF(T)\). Let \(X \subseteq PF(T)\). We define when a formula \(\varphi \in F(T)\) holds in \(X\); notation \(X \vdash \varphi\).

\[
\begin{align*}
X \vdash sRt & \text{ if } sRt \in X, \\
X \vdash Ps & \text{ if } Ps \in X, \\
X \vdash s \neg R & \text{ if } \forall t \in C(\Sigma) : sRt \notin X, \\
X \vdash \neg Ps & \text{ if } Ps \notin X.
\end{align*}
\]

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 [11] 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 (2.4)**

Let \( T = (\Sigma, D) \) be a term deduction system and let \( X \subseteq \text{PF}(T) \) 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 \( \varphi \) and all the instantiated hypotheses hold in \( X \). More formally: \( X \) agrees with \( T \) if

\[
\varphi \in X \iff \exists H/C \in D \text{ and } \sigma : V \rightarrow C(\Sigma) \text{ such that } \sigma(C) = \varphi \text{ and } \forall h \in H : X \vdash \sigma(h).
\]

**Definition (2.5)**

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

**Example (2.6)**

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 \( S(t \mapsto t') = n \) is a stratification (\( t' \) is a closed term).

**Definition (2.7)**

Let \( V \) be a set. If \( 0 \leq |V| < \aleph_0 \) we define \( d(V) = \omega_0 \). If \( |V| = \aleph_0 \) 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: \( d(H/C) = d(\text{PF}(H)) \). Let \( \omega_\alpha = \sup\{d(H/C) : H/C \in D\} \). The degree \( d(T) \) of a term deduction system \( T \) is \( \omega_0 \) if \( \alpha = 0 \) and \( \omega_{\alpha+1} \) otherwise.

**Example (2.8)**

It is easy to see that the degree of our running example is \( \omega_0 \).

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

**Definition (2.9)**

Let \( T = (\Sigma, D) \) be a term deduction system and let \( S : \text{PF}(T) \rightarrow \alpha \) be a stratification for an ordinal number \( \alpha \). We define a set \( T_S \subseteq \text{PF}(T) \) as follows.

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

We will frequently use unions over \( T_i^S \) and \( T_{i,j}^S \) in proofs; so, we introduce the following notations

\[
U_i^S = \bigcup_{i' \leq i} T_{i',j}^S (i \leq \alpha), \quad U_{i,j}^S = \bigcup_{i' \leq j} T_{i',j}^S (j \leq d(T)).
\]
We drop the sub- and superscripts $S$ and, for instance, render $U_i^S$ as $U_i$ and $T_S \vdash \varphi$ as $T \vdash \varphi$; provided no confusion arises. Now we define for all $i < \alpha$ and for all $j < d(T)$ the set $T_{i,j} = T_{i,j}^S$:

\[
T_{i,j} = \left\{ \varphi \mid S(\varphi) = i, \exists H/C \in D \text{ and } \sigma : Y \rightarrow C(\Sigma) \text{ with } \sigma(C) = \varphi, \forall h \in PF(H) : U_{i,j} \cup U_i \vdash \sigma(h) \text{ and } \forall h \in NF(H) : U_i \vdash \sigma(h) \right\}.
\]

The next theorem is taken from Verhoef [28] but its proof is essentially the same as a similar theorem of Groote [18].

**Theorem (2.10)**

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

**Example (2.11)**

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 (2.12)**

Let $T = (\Sigma, D)$ be a term deduction system with $D = D(T_p, T_i)$. Let in the following $K, L, M,$ and $N$ be index sets of arbitrary cardinality, let $s_k, t_i, v_m, t \in O(\Sigma)$ for all $k \in K, i \in L, m \in M,$ 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 \textit{pntyf} format if it has the form

\[
\{P_k s_k : k \in K\} \cup \{t_i R_l y_i : t \in L\} \cup \{\neg P_m u_m : m \in M\} \cup \{v_n = R_n : n \in N\}
\]

with $C = f(x_1, \ldots, x_n) R_t, f \in \Sigma$ 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 $\text{var}(d) = X \cup Y$ we call $d$ pure. A variable in $\text{var}(d)$ that does not occur in $X \cup Y$ is called free.

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

A deduction rule is in \textit{pntyf} format if it has the form above and $C = P f(x_1, \ldots, x_n)$ and it is in \textit{pntyz} format if it has the form above and $C = P x$ 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 \textit{panth} format, which stands for “predicates and \textit{nntyf)/\textit{nntyf} hybrid format”. A term deduction system is in \textit{panth} format if all its rules are. A term deduction system is called pure if all its rules are pure. The \textit{nntyf)/\textit{nntyf} format is defined in Groote [18].

**Example (2.13)**

The operational rules of our running example satisfy the \textit{panth} format and are pure.

**Definition (2.14)**

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 $x \rightarrow y$ is an edge of the variable dependency graph if and only if there is a positive relation $i R s \in E$ with $x \in \text{var}(i)$ and $y \in \text{var}(s)$.

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.
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 \( \Sigma_0 \) and \( \Sigma_1 \) be signatures. If for all \( f \in \Sigma_0 \cap \Sigma_1 \) the arity of \( f \) in \( \Sigma_0 \) is the same as the arity of \( f \) in \( \Sigma_1 \) then \( \Sigma_0 \oplus \Sigma_1 \), called the sum of \( \Sigma_0 \) and \( \Sigma_1 \), is the signature \( \Sigma_0 \cup \Sigma_1 \).

Example (3.2)
Let \( \Sigma_0 = \{ \_a : a \in A \} \cup \{ +, \cdot \} \) and \( \Sigma_1 = \{ \_d, +, \cdot \} \) be signatures. Then \( \Sigma_0 \oplus \Sigma_1 \) is defined and equals the signature of our running example. Moreover, \( \Sigma_0 \) 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 \( \Sigma_0 \) of (3.2) and the rules without a \( \_\rightarrow \) of table 1 and let \( T_1 \) have \( \Sigma_1 \) of (3.2) as its signature and the rules with \( \_\rightarrow \) 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.

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 [19] (without the adjective 'operational') where they used the first notion to characterize the completed trace congruence induced by their pure \( typt/tyxt \) format. Groote [18] gives the two definitions in the case that negative premises come into play. He used operational conservativity for a similar characterization result as in [19]. In Bol and Groote [12] the approach of Groote [18] is placed in a wider perspective. Aceto, Bloom and Vaandrager [1] 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 \( D = D(T^0, T^1) \). 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
\[
T_S \vdash s R t \iff T_{S^0} \vdash s R t
\]
and
\[
T_S \vdash P u \iff T_{S^0} \vdash P u
\]
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 \( T = (\Sigma, D) := T^0 \circ T^1 \) defined. If we have for all \( s, t \in C(\Sigma_0) \)
\[
s =^\varphi_\varphi t \iff s =^\varphi t
\]
we say that \( T \) is an operational conservative extension of \( T_0 \) up to \( \varphi \) equivalence; where \( \varphi \) is some semantic 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, \( \eta \) bisimulation equivalence, delay bisimulation equivalence, branching bisimulation equivalence, and more equivalences. We refer to Van Glabbeek’s linear time–branching time spectra [15] and [16] for more information on these equivalences. In the applications we make use of \( \epsilon \) bisimulation originating from Koymans and Vrancken [23]. 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 \( T = T^0 \circ T^1 \) be defined. If \( T \) is an operational conservative extension of \( T_0 \) then it is also an operational conservative extension up to \( \varphi \) equivalence; where \( \varphi \) 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 will 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 \( T_0 \circ T_1 \) is an operational conservative extension of \( T_0 \). The theorem is a threefold generalization of a similar result for Groote’s ntyft/ntyxt format [18]. 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 [19].

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 \( \Sigma_0 \) terms and a new relation or predicate symbol. Now if \( T = T^0 \circ T^1 \) 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 \( S : PF(T) \rightarrow \alpha \) be a stratification for \( T \) and let \( S^0 : PF(T^0) \rightarrow \alpha \) be the restriction of \( S \) to \( PF(T^0) \) (note that \( S^0 \) is a stratification).
Let \( u, w \in C(\Sigma_0), R \in Tr, P \in T_p, \) and \( v \in C(\Sigma) \). We are to show that the following two bi-implications hold

\[
T_s \vdash uRv \iff T_s^0 \vdash uRv,
\]

\[
T_s \vdash Pw \iff T_s^0 \vdash Pw.
\]

By definition (2.9) it suffices to prove the two bi-implications below for all \( i < \alpha \).

\[
T_i \vdash uRv \iff T_i^0 \vdash uRv \tag{1}
\]

\[
T_i \vdash Pw \iff T_i^0 \vdash Pw. \tag{2}
\]

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 (2.9) it suffices to prove for all \( j < d(T) \) that

\[
T_{i,j} \vdash uRv \implies T_{i,j}^0 \vdash uRv \tag{3}
\]

\[
T_{i,j} \vdash Pw \implies T_{i,j}^0 \vdash Pw. \tag{4}
\]

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 (2.9) there is a rule \( d \in D \)

\[
\{ P_{\sigma k} : k \in K \} \cup \{ t_i R_{iy} : i \in L \} \cup \{ \neg P_m u_m : m \in M \} \cup \{ v_n \neg R_n : n \in N \}
\]

with \( C = sRt \). There is also a closed substitution \( \sigma \) 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 \( \sigma(x) = X \) and \( Y = \{ y_i : i \in L \} \). Since \( d \) is pure we have that \( \forall r d = X \vee Y \). 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 \( y_i \in Y \) with \( \sigma(y_i) \in C(\Sigma) \setminus C(\Sigma_0) \). This contradicts the well-foundedness of the rule \( d \), for \( U_i \cup U_{i,j} \vdash \sigma(t_{i_0}) R_k \sigma(y_{i_0}) \) so with the induction hypotheses on \( i \) or \( j \) we find that \( \sigma(t_{i_0}) \in C(\Sigma) \setminus C(\Sigma_0) \). Since \( t_{i_0} \) is a \( \Sigma_0 \) term, this must be the result of a substitution. This can only be due to a variable \( y_i \in Y \).

With induction on the subscripts we find an infinite backward chain of edges \( y_{i_0} \leftarrow y_{i_1} \leftarrow \ldots \) 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 \( \Sigma_0 \) terms and a new relation or predicate symbol. By definition (2.9) 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 \( \Sigma_0 \) 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 \( U_i^0 \not\vdash \sigma(v_n \neg R_n) \). 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 (2.9) we have \( T_{i,j}^0 \vdash uRv \) so \( T_{i,j}^0 \vdash uRv \).

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 (2.9) it suffices to show for all \( j < d(T^0) \) that

\[
T_{i,j}^0 \vdash uRv \implies T_i \vdash uRv
\]

\[
T_{i,j}^0 \vdash Pw \implies T_i \vdash Pw.
\]

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 of (3.9).
A general conservative extension theorem . . . 5. Operational and equational conservativity

Example (3.10)

It is not hard to see that the term deduction system $T_0$ 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.

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 $(\Sigma_0 \oplus \Sigma_1, E_0 \cup E_1)$.

Example (3.12)

Let $\Sigma_0$ and $\Sigma_1$ 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 $(\Sigma_0, E_0)$ is the equational specification $BPA$. We define $DT$ to be the equational specification $(\Sigma_1, E_1)$. The equational specification $BPA_{dt}$ of our running example is just $BPA \oplus DT$.

<table>
<thead>
<tr>
<th>Equation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + y = y + x$</td>
<td>A1</td>
</tr>
<tr>
<td>$(x + y) + z = x + (y + z)$</td>
<td>A2</td>
</tr>
<tr>
<td>$x + z = x$</td>
<td>A3</td>
</tr>
<tr>
<td>$(x + y)z = xz + yz$</td>
<td>A4</td>
</tr>
<tr>
<td>$(zy)z = z(yz)$</td>
<td>A5</td>
</tr>
<tr>
<td>$\sigma_d(x) + \sigma_d(y) = \sigma_d(x + y)$</td>
<td>DT1</td>
</tr>
<tr>
<td>$\sigma_d(x) \cdot y = \sigma_d(x \cdot y)$</td>
<td>DT2</td>
</tr>
</tbody>
</table>

Table 2. The axioms of $BPA$ and $DT$.

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 $\varphi$ 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 $\varphi$ equivalence model induced by $T_0$ and let $E$ be a sound axiomatization with respect to the $\varphi$ equivalence model induced by $T$. If $T$ is an operational conservative extension of $T_0$ up to $\varphi$ 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 $\varphi$ equivalence model induced by $T$.
A general conservative extension theorem . . . : 4. Ground confluence modulo equations

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 \( s =_E^0 t \). Since \( T \) is an operational conservative extension of \( T_0 \) up to \( \varphi \) equivalence we have that \( s =\varphi^0 t \). Since \( E_0 \) is complete with respect to \( \varphi \) 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 \( \varphi \) equivalent \( \Sigma \) 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 \( \varphi \) equivalent in \( T \) and since this is an operational conservative extension of \( T_0 \) up to \( \varphi \) equivalence, we find that \( s' \) and \( t' \) are \( \varphi \) 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 [7]. Baeten and Bergstra [5] 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_{dt} = BPA + 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 \((\Sigma, R)\) with \(\Sigma\) a signature and \(R\) a set of rewrite rules. Rewrite rules are pairs of terms (over \(\Sigma\)) that we denote \( s \rightarrow t \). We suppose that \( s \) is not a variable and that \( \text{var}(t) \subseteq \text{var}(s) \). The one step rewrite relation \( \rightarrow_1 \) is the smallest relation on terms containing \( R \) that is closed under substitutions and contexts. The rewrite relation \( \rightarrow_R \) is the transitive-reflexive closure of the one step rewrite relation \( \rightarrow_1 \). Often, we refer to a term rewriting system \((\Sigma, R)\) 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 \( \rightarrow_{R/E} \) is defined as \( =_E \circ \rightarrow_1 \circ =_E \). The rewriting relation \( \rightarrow_{R/E} \) is the transitive-reflexive closure of the one step rewrite relation \( \rightarrow_{R/E} \). 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 \( \rightarrow \) be the rewriting relation \( \rightarrow_{R/E} \). Let \( s \) be a term. If for all \( s_0, s_1 \) such that \( s \rightarrow s_0 \) and \( s \rightarrow s_1 \) there is a term \( s' \) such that \( s_0 \rightarrow s' \) and \( s_1 \rightarrow s' \) we say that the rewriting relation \( \rightarrow_{R/E} \) is Church-Rosser or confluent. See figure 1. We call \( \rightarrow_{R/E} \) ground Church-Rosser if it is Church-Rosser for closed terms. Sometimes, we will write CR instead of Church-Rosser. We also say that \( \rightarrow \) is Church-Rosser (or confluent) modulo \( E \); we write CR/E. In the literature we also see E-Church-Rosser and E-confluence if \( \rightarrow_{R/E} \) is confluent in the above sense; see, for instance, Jouannaud and Muñoz [21].
A general conservative extension theorem ... 4. Ground confluence modulo equations

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 \( \rightarrow_{R/E} \) is ground CR\( ^E \) if for all ground terms \( s \) and \( t \) such that \( s = t \) there are terms \( s' \) and \( t' \) such that \( s \rightarrow_{R/E} s' \), \( t \rightarrow_{R/E} t' \), and \( s' = E t' \).

Remark (4.5)

It is easily seen that \( \rightarrow_R \) is ground CR\( ^E/E \) if and only if it is ground CR\( ^E \).

Definition (4.6)

A term rewriting system \( R \) is terminating if there exists no infinite sequence \( s_0 \rightarrow_R s_1 \rightarrow_R s_2 \ldots \).

We call a term \( s \) a normal form if we do not have \( s \rightarrow_R s' \) for any \( s' \).

Theorem (4.7)

Let \( L = (\Sigma, E) \) 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 a set \( R^- = (E \setminus A_0) \cup R_0^- \) into a set of rewrite rules \( R \). Suppose that \( \rightarrow_R \) is terminating and that normal forms are \( \Sigma_0 \) terms (so \( L \) has the elimination property). If \( \rightarrow_{R_0/A_0} \) is ground Church-Rosser then \( \rightarrow_{R/A_0} \) is also ground Church-Rosser.

Proof. Let \( s \) and \( t \) be ground \( \Sigma \) terms and suppose that \( E \vdash s = t \). By assumption, there are ground \( \Sigma_0 \) terms \( s' \) and \( t' \) with \( s \rightarrow_R s' \) and \( t \rightarrow_R t' \). So \( E \vdash s' = t' \). Since \( L \) is a conservative extension of \( L_0 \) we now have that \( E_0 \vdash s' = t' \). Since \( \rightarrow_{R_0/A_0} \) is ground CR there are \( s_0 \) and \( t_0 \) such that \( s' \rightarrow_{R_0/A_0} s_0 \), \( t' \rightarrow_{R_0/A_0} t_0 \), and \( A_0 \vdash s_0 = t_0 \). Since \( R_0 \subseteq R \) we also have \( s' \rightarrow_{R/A_0} s_0 \). Since \( s \rightarrow_R s' \) we also have \( s \rightarrow_{R/A_0} s' \) (simply put \( s = A_0 \), \( s_0, \ldots, s' = A_0 \) between the one step rewritings). So we find that \( s \rightarrow_{R/A_0} s_0 \). In the same way we find that \( t \rightarrow_{R/A_0} t_0 \), and we have \( A_0 \vdash s_0 = t_0 \). This implies using remark (4.5) that \( \rightarrow_{R/A_0} \) 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. Note-worthy perhaps, is that we could not find any 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.

The operational approach

Moller and Tofts [25] 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.

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 2. And we will treat some typical cases more elaborately. We note that the examples in figure 2 contain both known results and new results.

In the introduction we mentioned the problems concerning the confluence of ACP that Bergstra and Klop [10] 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 \( \bullet \)-labelled arrow from ACP to BPA\(_{\text{d}}\) in figure 2. We show that all our general results apply to this arrow.

Van Glabbeek [14] gives an operational semantics for Bergstra and Klop's ACP [10] and for their sequential subsystem BPA\(_{d}\) [10]. 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 [7], for instance, show that BPA\(_{d}\) 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\(_{d}\). 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\(_{d}\). The associated term rewriting system of BPA\(_{d}\) consists of five rewrite rules and two equations, which is a considerable reduction since the term rewriting system for ACP has many more rules.

\[\begin{array}{c}
\text{ACP} \\
\text{BPA} \\
\text{BPA}_{d} \\
\end{array} \quad \begin{array}{c}
\text{PA} \\
\text{PA}_{d} \\
\end{array} \quad \begin{array}{c}
\text{ATP} \\
\text{ATP}_{u} \\
\text{ATP}_{w} \\
\end{array}
\]

Figure 2. Some applications.

Now, we discuss figure 2. An arrow \( A \rightarrow B \) 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 [9]. The acronym ASP stands for the algebra of sequential processes. This system stems from Milner [24]. 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 2.

Let \( x = y \) be one of PR, RN, \( \lambda \), \( \Delta \), \( \theta \), or a combination of them. The abbreviations stand for projections, renamings, simple state operators, extended state 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 textbook of Baeten and Weijland [7]. 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 the complete BPA or the complete BPA\( \delta \) (if the extension contains already a \( \delta \)).

Now, let \( x = y \) be Milner's silent action \( \tau \). We already mentioned in the introduction that systems containing the three \( \tau \) laws of Milner have in general bad rewriting properties. The conservativity of ACP\( _\tau \) over ACP was proved semantically by Bergstra and Klop [8] since the second and third \( \tau \) 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\( _\tau \) is just the one of ACP but now \( \alpha \) ranges also over \( \tau \) itself. It is easy to see that the conditions of theorem (3.9) are satisfied, so ACP\( _\tau \) is an operational conservative extension of ACP. Now with theorem (3.8) we find that ACP\( _\tau \) is an operational conservative extension up to rooted \( \tau \) bisimulation equivalence of ACP. We should note that the definition of (rooted) \( \tau \) bisimulation equivalence in the presence of the predicates \( \neg \omega \rightarrow \sqrt{\cdot} \) as given by Baeten and Weijland [7] is wrong, since the first \( \tau \) law of Milner is not sound. This can be easily repaired. Since ACP\( _\tau \) is sound and complete and since ACP\( _\tau \) is sound with respect to this equivalence, we find with theorem (3.14) that ACP\( _\tau \) 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 BPA\( \delta \), we find their completeness with the aid of the completeness of BPA\( \delta \). The systems have bad term rewriting properties so the ground confluence results do 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 \( \tau \). 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\( _\tau \) is an operational conservative extension up to branching bisimulation equivalence [17] of ACP. Also in this case we note that the definition of branching bisimulation equivalence in the presence of the predicates \( \neg \omega \rightarrow \sqrt{\cdot} \) as given by Baeten and Weijland [7] is wrong, since the first \( \tau \) law of Milner is not sound. Having repaired this definition, we find in the same way as above that ACP with the branching \( \tau \) axioms [17], denoted ACP\( _\tau \), 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\( _\tau \) we find the completeness for them with the completeness of BPA\( _\tau \). The branching \( \tau \) axioms have better term rewriting properties [3] than the \( \tau \) laws of Milner (that we discussed above). So our ground confluence result may be useful, as well.

Let \( x = y \) be the empty process \( \varepsilon \) of Koymans and Vrancken [23]; see also Vrancken [29]. We can show operational and equational conservativity for all arrows from a system with an \( \varepsilon \) to a
A general conservative extension 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

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. When a counterexample is found it may be worthwhile to adapt our theory so that it can deal with the counterexample in a satisfactory way.

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.

7. References


A general conservative extension theorem... References

[23] C. P. J. Koomans, J. L. M. Vrancken, Extending process algebra with the empty process e, Logic Group Preprint Series Nr. 1, Department of Philosophy, Utrecht University, 1985.
In this series appeared:


91/02 R.P. Nederpelt  
H.C.M. de Swart  Implication. A survey of the different logical analyses "if...,then...", p. 26.

91/03 J.P. Katoen  
L.A.M. Schoenmakers  Parallel Programs for the Recognition of P-invariant Segments, p. 16.

91/04 E. v.d. Sluis  

91/05 D. de Reus  An Implementation Model for GOOD, p. 18.

91/06 K.M. van Hee  SPECIFICATIEMETHODEN, een overzicht, p. 20.

91/07 E.Poll  CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.


91/10 R.C.Backhouse  
P.J. de Bruin  
P. Hoogendijk  
G. Malcolm  
E. Voermans  

91/11 R.C. Backhouse  
P.J. de Bruin  
P. Hoogendijk  
G. Malcolm  
E. Voermans  
J. van der Woude  Relational Catamorphism, p. 31.

91/12 E. van der Sluis  A parallel local search algorithm for the travelling salesman problem, p. 12.


91/14 P. Lemmens  The PDB Hypermedia Package. Why and how it was built, p. 63.

91/15 A.T.M. Aerts  

91/16 A.J.J.M. Marcelis  An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17 A.T.M. Aerts  
P.M.E. de Bra  
<table>
<thead>
<tr>
<th>Page</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/18</td>
<td>Rik van Geldrop</td>
<td>Transformational Query Solving, p. 35.</td>
</tr>
<tr>
<td>91/19</td>
<td>Erik Poll</td>
<td>Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.</td>
</tr>
<tr>
<td>91/23</td>
<td>K.M. van Hee, L.J. Somers, M. Voorhoeve</td>
<td>Z and high level Petri nets, p. 16.</td>
</tr>
<tr>
<td>91/24</td>
<td>A.T.M. Aerts, D. de Reus</td>
<td>Formal semantics for BRM with examples, p. 25.</td>
</tr>
<tr>
<td>91/25</td>
<td>P. Zhou, J. Hooman, R. Kuiper</td>
<td>A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.</td>
</tr>
<tr>
<td>91/27</td>
<td>F. de Boer, C. Palamidessi</td>
<td>Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.</td>
</tr>
<tr>
<td>91/28</td>
<td>F. de Boer</td>
<td>A compositional proof system for dynamic process creation, p. 24.</td>
</tr>
<tr>
<td>91/30</td>
<td>I.C.M. Baeten, F.W. Vaandrager</td>
<td>An Algebra for Process Creation, p. 29.</td>
</tr>
<tr>
<td>91/33</td>
<td>W. v.d. Aalst</td>
<td>The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.</td>
</tr>
<tr>
<td>91/34</td>
<td>J. Coenen</td>
<td>Specifying fault tolerant programs in deontic logic, p. 15.</td>
</tr>
<tr>
<td>Number</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>92/01</td>
<td>J. Coenen, J. Zwiers, W.-P. de Roever</td>
<td>A note on compositional refinement, p. 27.</td>
</tr>
<tr>
<td>92/02</td>
<td>J. Coenen, J. Hooman</td>
<td>A compositional semantics for fault tolerant real-time systems, p. 18.</td>
</tr>
<tr>
<td>92/03</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Real space process algebra, p. 42.</td>
</tr>
<tr>
<td>92/05</td>
<td>J.P.H.W.v.d.Eijnde</td>
<td>Conservative fixpoint functions on a graph, p. 25.</td>
</tr>
<tr>
<td>92/06</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Discrete time process algebra, p.45.</td>
</tr>
<tr>
<td>92/07</td>
<td>R.P. Nederpelt</td>
<td>The fine-structure of lambda calculus, p. 110.</td>
</tr>
<tr>
<td>92/10</td>
<td>P.M.P. Rambags</td>
<td>Composition and decomposition in a CPN model, p. 55.</td>
</tr>
<tr>
<td>92/13</td>
<td>F. Kamareddine</td>
<td>Set theory and nominalisation, Part II, p.22.</td>
</tr>
<tr>
<td>92/14</td>
<td>J.C.M. Baeten</td>
<td>The total order assumption, p. 10.</td>
</tr>
<tr>
<td>92/15</td>
<td>F. Kamareddine</td>
<td>A system at the cross-roads of functional and logic programming, p.36.</td>
</tr>
<tr>
<td>92/16</td>
<td>R.R. Seljée</td>
<td>Integrity checking in deductive databases; an exposition, p.32.</td>
</tr>
<tr>
<td>92/17</td>
<td>W.M.P. van der Aalst</td>
<td>Interval timed coloured Petri nets and their analysis, p. 20.</td>
</tr>
<tr>
<td>92/18</td>
<td>R.Nederpelt, F. Kamareddine</td>
<td>A unified approach to Type Theory through a refined lambda-calculus, p. 30.</td>
</tr>
<tr>
<td>92/20</td>
<td>F.Kamareddine</td>
<td>Are Types for Natural Language? P. 32.</td>
</tr>
<tr>
<td>92/21</td>
<td>F.Kamareddine</td>
<td>Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.</td>
</tr>
<tr>
<td>Year/Issue</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>92/22</td>
<td>R. Nederpelt, F. Kamareddine, E. Klein</td>
<td>A useful lambda notation, p. 17.</td>
</tr>
<tr>
<td>92/23</td>
<td>F. Kamareddine, E. Klein</td>
<td>Nominalization, Predication and Type Containment, p. 40.</td>
</tr>
<tr>
<td>92/24</td>
<td>M. Codish, D. Dams, Eyal Yardeni</td>
<td>Bottom-up Abstract Interpretation of Logic Programs, p. 33.</td>
</tr>
<tr>
<td>92/25</td>
<td>E. Poll</td>
<td>A Programming Logic for Fω, p. 15.</td>
</tr>
<tr>
<td>93/01</td>
<td>R. van Geldrop</td>
<td>Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.</td>
</tr>
<tr>
<td>93/02</td>
<td>T. Verhoeff</td>
<td>A continuous version of the Prisoner’s Dilemma, p. 17</td>
</tr>
<tr>
<td>93/03</td>
<td>T. Verhoeff</td>
<td>Quicksort for linked lists, p. 8.</td>
</tr>
<tr>
<td>93/04</td>
<td>E.H.L. Aarts, J.H.M. Korst, P.J. Zwietering</td>
<td>Deterministic and randomized local search, p. 78.</td>
</tr>
<tr>
<td>93/05</td>
<td>J.C.M. Baeten, C. Verhoeff</td>
<td>A congruence theorem for structured operational semantics with predicates, p. 18.</td>
</tr>
<tr>
<td>93/06</td>
<td>J.P. Veltkamp</td>
<td>On the unavoidability of metastable behaviour, p. 29</td>
</tr>
<tr>
<td>93/07</td>
<td>P.D. Moerland</td>
<td>Exercises in Multiprogramming, p. 97</td>
</tr>
<tr>
<td>93/08</td>
<td>J. Verhoosel</td>
<td>A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.</td>
</tr>
<tr>
<td>93/10</td>
<td>K.M. van Hee</td>
<td>Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.</td>
</tr>
<tr>
<td>Page</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>93/16</td>
<td>H. Schepers, J. Hoorman</td>
<td>A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27</td>
</tr>
<tr>
<td>93/17</td>
<td>D. Alstein, P. van der Stok</td>
<td>Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.</td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.</td>
</tr>
<tr>
<td>93/19</td>
<td>G-J. Houben</td>
<td>The Design of an Online Help Facility for ExSpect, p.21.</td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
<td>A Typechecker for Bijective Pure Type Systems, p. 28.</td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
<td>Relational Algebra and Equational Proofs, p. 23.</td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll and Paula Severi</td>
<td>Pure Type Systems with Definitions, p. 38.</td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
<td>Multi-dimensional Petri nets, p. 25.</td>
</tr>
<tr>
<td>93/27</td>
<td>T. Kloks and D. Kratsch</td>
<td>Finding all minimal separators of a graph, p. 11.</td>
</tr>
<tr>
<td>93/28</td>
<td>F. Kamareddine and R. Nederpelt</td>
<td>A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.</td>
</tr>
<tr>
<td>93/29</td>
<td>R. Post and P. De Bra</td>
<td>GOLD, a Graph Oriented Language for Databases, p. 42.</td>
</tr>
<tr>
<td>93/30</td>
<td>J. Deogun, T. Kloks, D. Kratsch, H. Müller</td>
<td>On Vertex Ranking for Permutation and Other Graphs, p. 11.</td>
</tr>
<tr>
<td>93/31</td>
<td>W. Körver</td>
<td>Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.</td>
</tr>
<tr>
<td>93/33</td>
<td>L. Loyens and J. Moonen</td>
<td>ILIAS, a sequential language for parallel matrix computations, p. 20.</td>
</tr>
<tr>
<td>No.</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>93/34</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Real Time Process Algebra with Infinitesimals, p.39.</td>
</tr>
<tr>
<td>93/36</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Non Interleaving Process Algebra, p. 17.</td>
</tr>
</tbody>
</table>