Formalizing the Confluence of Orthogonal Rewriting Systems

[1]Ana Cristina Rocha Oliveira and [1,2]Mauricio Ayala-Rincón
Grupo de Teoria da Computação, Departamentos de [1]Matemática e [2]Computação
Universidade de Brasília
Brasília D.F., Brazil
Email: `a.c.r.oliveira@mat.unb.br, ayala@unb.br`

Orthogonality is a discipline of programming that in a syntactic manner guarantees determinism of functional specifications. Essentially, orthogonality avoids, on the one side, the inherent ambiguity of non determinism, prohibiting the existence of different rules that specify the same function and that may apply simultaneously (*non-ambiguity*), and, on the other side, it eliminates the possibility of occurrence of repetitions of variables in the left-hand side of these rules (*left linearity*). In the theory of term rewriting systems (TRSs) determinism is captured by the well-known property of confluence, that basically states that whenever different computations or simplifications from a term are possible, the computed answers or the obtained reduced terms should coincide. Although the proof is technically elaborated, confluence is well-known to be a consequence of orthogonality. Thus, orthogonality is an important mathematical discipline intrinsic to the specification of recursive functions that is naturally applied in functional programming and specification. Starting from a formalization of the theory of TRSs in the proof assistant PVS, this work describes how confluence of orthogonal TRSs is being formalized in this proof assistant. Substantial progress has been done in this research, obtaining until now complete formalizations for some similar, but restricted properties, such as a complete formalization for the property of confluence of non-ambiguous and (left and right) linear TRSs.

1 Introduction

Termination and confluence of term rewriting systems (TRSs) are well-known undecidable properties that are related with termination of computer programs and determinism of their outputs. Under the hypothesis of termination, confluence is guaranteed by the critical pair criterion of Knuth-Bendix(-Huet) ([8] + [7]), which establishes that whenever all critical pairs of a given terminating rewriting system are joinable, the system is confluent. This criterion was fully formalized in the proof assistant PVS in [6] over the PVS *theory* `trs` [5], that is available in the NASA LaRC PVS library [12]. Without termination, confluence analysis results more complex, but several programming disciplines, from which one could remark orthogonality, guarantee confluence without the necessity of termination.

In the context of the theory of recursive functions and functional programming as in the one of TRSs, the programming discipline of orthogonality follows two restrictions:

- *left-linearity*
- *non-ambiguity*

The former restriction, allows only definitions or rules in which each variable may appear only once on the left-hand side (**lhs**, for short) of each rule; the latter restriction, avoids the inclusions of definitions or rules that could simultaneously apply.

A few examples are given to illustrate these notions and the necessity of both restrictions, left-linearity and non-ambiguity, in order to guarantee confluence. Below, it is presented a simple example of an orthogonal TRS for the factorial function.

$$(R_1)\ factorial(n+1) \rightarrow (n+1)\times factorial(n)$$
$$(R_2)\ factorial(0) \rightarrow 1$$

Indeed, the rules do not overlap and the variable n occurs only once on the lhs of the rule R_1 and none on the lhs of the rule R_2. Intuitively, it is clear that this specification allows deterministic computations of the factorial of each possible natural number given as input.

The TRS below does not own the same properties. In fact, it is neither left linear nor ambiguous.

$$\begin{array}{ll} (Q_1) & (a+b)^2 \rightarrow a^2+2ab+b^2 \\ (Q_2) & a+a \rightarrow 2a \end{array}$$

This TRS is ambiguous, because its rules overlap giving the divergence illustrated below.

$$\begin{tikzcd} & (a+a)^2 \arrow[dl, "Q_1"'] \arrow[dr, "Q_2"] & \\ a^2+2aa+a^2 & & (2a)^2 \end{tikzcd}$$

Namely, one cannot join the terms of this divergence, $(a^2+2aa+a^2)$ and $(2a)^2$. Also, note that Q_2 is a non left-linear rule, besides the fact that Q_1 and Q_2 overlap.

The TRS presented below does not own the same properties. In fact, it is left linear, but ambiguous.

$$(Q) \quad f(f(x)) \rightarrow g(x)$$

This TRS is ambiguous because its unique rule overlaps with itself giving the divergence illustrated below.

$$\begin{tikzcd} & f(f(f(a))) \arrow[dl, "Q"'] \arrow[dr, "Q"] & \\ g(f(a)) & & f(g(a)) \end{tikzcd}$$

Namely, one cannot join the terms of this divergence, $g(f(a))$ and $f(g(a))$ because they are irreducible.

However, non-ambiguity is not sufficient to guarantee confluence. See the TRS in the next example.

$$\begin{array}{ll} (S_1) & f(x,x) \rightarrow c \\ (S_2) & f(x,g(x)) \rightarrow b \\ (S_3) & a \rightarrow g(a) \end{array}$$

It can be checked that this TRS is non-ambiguous, but the variable x appears twice in the lhs of the rules (S_1) and (S_2). Now, observe the divergence below.

$$\begin{tikzcd} & f(a,a) \arrow[dl, "S_1"'] \arrow[dr, "S_3"] & \\ c & & f(a,g(a)) \arrow[d, "S_2"] \\ & & b \end{tikzcd}$$

Once more we have two different irreducible terms.

In this work we report a formalization of the property of confluence of orthogonal systems in the proof assistant PVS. The formalization uses the PVS theory `trs`, but several additional notions such as the one of parallel rewriting were included in order to follow the standard proof approach of this property that is based on the proof of the diamond property for the parallel reduction associated to any orthogonal TRS. In the current state of this formalization, several technical details that are related with properties of terms and subterms involved in one-step of parallel reduction are axiomatized. Additionally, the PVS theory includes a complete formalization of the confluence of non-ambiguous and linear TRS. For the benefit of the reviewing process, the whole theory, in its current status, is available in the page `www.mat.unb.br/∼ayala/publications.html`.

2 Basic Notions and Definitions

Standard notation of of the theory of rewriting is used as in [3] or [4]: terms are represented as trees and positions of a term t as sequences of naturals indexing paths from the root of the tree to the subterms (sub tree nodes). For a position π of a term t, $t|_\pi$ denotes the subterm at position π and parallel positions π and π' are sequences such that neither π is a prefix of π' nor π' is a prefix of π'. Given a TRS R, the rewriting relation is denoted as $\to_R$, and R is omitted when it is clear from the context. Composition of relations is denoted as $\circ$. The inverse of $\to$ is denoted by $\leftarrow$ and syntactic equality by $=$. Then, the reflexive and the symmetric closure of the relation $\to$ are given by $\to \cup =$ and $\leftarrow \cup \to$, respectively. For brevity, the former is denoted as $\to^=$ and the latter as $\leftrightarrow$. The reflexive transitive and the equivalence closure of $\to$ are denoted as $\to^*$ and $\leftrightarrow^*$, respectively. Similarly, ${}^*\!\!\leftarrow$ will denote the reflexive transitive closure of the inverse of $\to$. The relations of local divergence, divergence and juntability are given by $\leftarrow \circ \to$, ${}^*\!\!\leftarrow \circ \to^*$ and $\to^* \circ {}^*\!\!\leftarrow$, respectively.

One says that $\to$ is

- *confluent* whenever $({}^*\!\!\leftarrow \circ \to^*) \subseteq (\to^* \circ {}^*\!\!\leftarrow)$,
- *triangle-joinable* if $(\leftarrow \circ \to) \subseteq (\to \circ {}^=\!\!\leftarrow) \cup (\to^= \circ \leftarrow)$, and
- that $\to$ has the *diamond property* if $(\leftarrow \circ \to) \subseteq (\to \circ \leftarrow)$.

Confluence means that any divergence can be joined; triangle-joinability means that any local divergence can be joined being that the joinability needs only one reduction or none on one of the sides and; diamond property means that each local divergence can be joined in exaclty one step on each side of the divergence.

A well-defined set of terms is built from a given signature and a set of enumerable variables. A rule $e = (l, r)$ is a pair of terms such that the first one cannot be a variable and the variables occurring in the second one should appear in the first one. A TRS is given as a set of rules. The reduction relation $\to_E$ induced by a TRS E is built as follows (see Fig. 1): a term t reduces to t_0 (denoted as $t \to t_0$, for short) if there are a position π from t, a rule $e \in E$ and a substitution σ such that:

- $t|_\pi = lhs(e)\sigma$, i.e., the subterm of t at position π is the lhs of the rule e instantiated by the substitution σ;
- and t_0 is obtained from t, by replacing the subterm at position π as $t_0|_\pi = rhs(e)\sigma$, i.e., the subterm at position π, that is the σ instance of the lhs of the rule e, $lhs(e)\sigma$, is replaced by the σ instance of the right-hand side (**rhs**, for short) of the rule, $rhs(e)\sigma$.
- the only change done in order to obtain t_0 from t, occurs at the position π.

All this is summarized by the following notation:

$$t \;=\; t[\pi \leftarrow lhs(e)\sigma] \quad \rightarrow_E \quad t[\pi \leftarrow rhs(e)\sigma] \;=\; t_0,$$

where, in general, $u[\pi \leftarrow v]$ denotes the term obtained from u by replacing the subterm at position π of u by the term v.

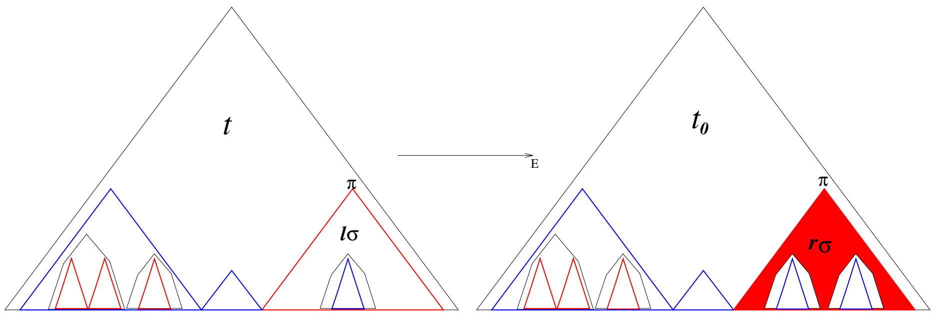

Figure 1: $t \rightarrow t_0$ ($l = lhs(e)$ and $r = rhs(e)$)

As previously mentioned, another relation that is used to prove the theorem of confluence of orthogonal TRSs is *parallel reduction*: given terms t_1 and t_2, one says that t_1 reduces in parallel to t_2, denoted as $t_1 \rightrightarrows t_2$ (see Fig. 3), whenever there exist finite sequences $\Pi := \pi_1, \ldots, \pi_n$; $\Sigma := \sigma_1, \ldots, \sigma_n$ and $\Gamma := e_1, \ldots, e_n$ of parallel positions of t_1, substitutions and rules, respectively, such that:

- $t_1|_{\pi_i} = lhs(e_i)\sigma_i$, i.e., the subterm of t_1 at position π_i is the lhs of the rule e_i instantiated by the substitution σ_i;
- and t_2 is obtained from t_1, by replacing all subterms at positions in Π as $t_2|_{\pi_i} = rhs(e_i)\sigma_i$, i.e., for all i, the subterm at position π_i, that is the σ_i instance of the lhs of the rule e_i, $lhs(e_i)\sigma_i$, is replaced by the σ_i instance of the rhs of the rule, $rhs(e_i)\sigma_i$.
- the only changes done in order to obtain t_2 from t_1, occur at the positions in Π.

All this is summarized by the following notation:

$$t_1 \;=\; t_1[\pi_1 \leftarrow l_1\sigma_1]\ldots[\pi_n \leftarrow l_n\sigma_n] \quad \rightrightarrows_E \quad t_1[\pi_1 \leftarrow r_1\sigma_1]\ldots[\pi_n \leftarrow r_n\sigma_n] \;=\; t_2,$$

where, $l_i = lhs(e_i)$ and $r_i = rhs(e_i)$, for $1 \leq i \leq n$.

3 Specification of Orthogonality in PVS

The previously mentioned complete formalization of the theory of TRSs in PVS, presented in [5] and called the PVS *theory* `trs`, includes all necessary basic notions and proved properties in order to formalize elaborated theorems of the theory of rewriting such as the one of confluence of orthogonal systems. The *theory* `trs` includes specifications and formalizations of the algebra of terms, sub terms and positions, properties of abstract reduction systems, confluence and termination, among others. The development of a PVS *theory* called `orthogonality` is in progress and it deals specifically with orthogonality definitions, properties, lemmas and theorems. Among the definitions specified inside the *theory* `orthogonality` one could mention the basic boolean ones listed below.

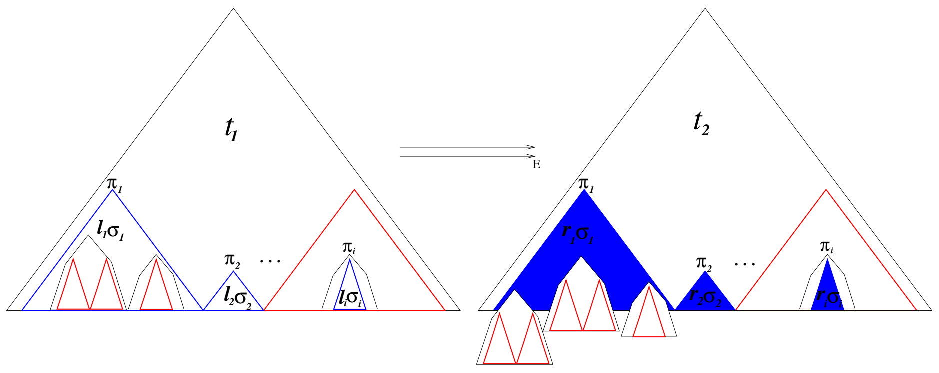

Figure 2: $t_1 \Longrightarrow t_2$ ($l_i = lhs(e_i)$ and $r_i = rhs(e_i)$)

```
- Ambiguous?(E): bool =  EXISTS (t1, t2) : CP?(E)(t1,t2)

- linear?(t): bool = FORALL (x | member(x,Vars(t))) : Card[position](Pos_var(t,x)) = 1

- Right_Linear?(E): bool = FORALL (e1 | member(e1, E)) : linear?(rhs(e1))

- Left_Linear?(E): bool = FORALL (e1 | member(e1, E)) : linear?(lhs(e1))

- Linear?(E): bool = Left_Linear?(E) AND Right_Linear?(E)

- Orthogonal?(E): bool =  Left_Linear?(E) AND NOT Ambiguous?(E)
```

In these definitions `E` is a set of rewriting rules (equations).

- Ambiguity (`Ambigous?`) of a TRS is defined as the existence of critical pairs `t1` and `t2` (`CP?(E)(t1,t2)`) produced by the rewriting rules in `E`.
- A term `t` is said to be linear (`linear?`), whenever each variable `x` that occurs in `t`, `Var(t)`, occurs only in a position of `t`. `member(a, A)` denotes that `a` belongs to the set `A`.
- A TRS `E` is said to be right or left linear (resp., `Right_Linear?` or `Left_Linear?`), whenever all right- or left-hand sides of the rules in `E` are linear, respectively.
- A TRS `E` is linear (`Linear?`), whenever it is left and right linear.
- A TRS `E` is orthogonal, whenever it is left linear and non ambiguous.

More elaborated auxiliary definitions are specified as:

```
- local_joinability_triangle?(R) : bool = FORALL(t, t1, t2) : R(t, t1) & R(t, t2) =>
             EXISTS s : (RC(R)(t1, s) & R(t2, s)) OR (R(t1, s) & RC(R)(t2, s))

- replaceTerm(s: term, t: term, (p: positions?(s))): RECURSIVE term =
     IF length(p) = 0  THEN t
      ELSE LET st = args(s), i = first(p), q = rest(p),
           rst = replace(replaceTerm(st(i-1), t, q), st,i-1)
           IN app(f(s), rst)
      ENDIF MEASURE length(p)

- reduction?(E)(s,t): bool =  EXISTS ( (e | member(e, E)), sigma, (p: positions?(s))):
                        subtermOF(s, p) = ext(sigma)(lhs(e))  &  t = replaceTerm(s, ext(sigma)(rhs(e)), p)
```

```
- replace_par_pos(s, (fsp : SPP(s)), fse | fse`length = fsp`length, fss  | fss`length = fsp`length)
    RECURSIVE term =
     IF length(fsp) = 0 THEN s
     ELSE replace_par_pos(replaceTerm(s, ext(fss(0))(rhs(fse(0))), fsp(0)), rest(fsp), rest(fse), rest(fss))
       ENDIF MEASURE length(fsp)

- parallel_reduction?(E)(s,t): bool =
   EXISTS (fsp: SPP(s), fse | (FORALL (i : below[fse`length]) : member(fse`seq(i), E)), fss) :
           fsp`length = fse`length AND fsp`length = fss`length
                 AND (FORALL (i : below[fsp`length]) : subtermOF(s, fsp(i)) = ext(fss(i))(lhs(fse(i))))
                 AND t = replace_par_pos(s, fsp, fse, fss)
```

- If a TRS has the property `local_joinability_ triangle?(R)`, then every one-step divergence can be joined in one step, on one hand, and in one or zero steps, on the other hand, i.e., it is *triangle-joinable*. `RC(R)` is the reflexive closure of the rewriting relation `R`.
- To change the subterm of `s` at position `p` by the term `t`, we can use the function `replaceTerm(s, t, p)`.
- `reduction?(E)` is the rewriting relation based on the TRS `E`. It draws a necessary rule `e`, substitution `sigma` and position `p` from `s` to reduce `s` to `t` since `t` is exactly `replaceTerm(s, ext(sigma)(rhs(e)), p)`. Analitically, it is unnecessary to discriminate between a substitution σ, that is a map from variables into terms, and its homeomorphic extension, that is, its extension to a function from terms to terms. But computationally, the construction is necessary and here it is done by the operator `ext`.
- The changes done by $\rightrightarrows$ are specified through the `replace_par_pos` recursive function, whose parameters are: `s`, a term; `fsp`, a finite sequence of positions of `s`; `fss`, a sequence of substitutions and `fse`, a sequence of rewrite rules, such that `length(fsp) = length(fss) = length(fse)`.
- `parallel_reduction?` is the relation $\rightrightarrows$ itself, assuming the existence of the necessary finite sequences `fsp`, `fss` and `fse`, for the term `s`, as in the previous item, and that the term `t` is exactly the result of `replace_par_pos` applied to these parameters.

The main lemmas and theorems specified about orthogonality are presented below. All presented lemmas were formalized.

The lemma `Linear_and_Non_ambiguous_implies_ confluent` is a weaker version of the lemma of confluence of Orthogonal TRSs that is the last one.

```
- Linear_and_Non_ambiguous_implies_triangle: LEMMA
  FORALL (E) : Linear?(E) AND NOT Ambiguous?(E) IMPLIES local_joinability_triangle?(reduction?(E))

- One_side_diamond_implies_conflent: LEMMA
  local_joinability_triangle?(R) IMPLIES confluent?(R)

- Linear_and_Non_ambiguous_implies_confluent: LEMMA
  FORALL (E) : ((Linear?(E) AND NOT Ambiguous?(E) ) IMPLIES confluent?(reduction?(E)))

- parallel_reduction: LEMMA
  (reduction?(E)(t1, t2) => parallel_reduction?(E)(t1, t2))
   & (parallel_reduction?(E)(t1, t2) => RTC(reduction?(E))(t1, t2))

- parallel_reduction_is_DP: LEMMA
  Orthogonal?(E) => diamond_property?(parallel_reduction?(E))
```

```
- Orthogonal_implies_confluent: LEMMA
  FORALL (E : Orthogonal) : LET RRE = reduction?(E) IN confluent?(RRE)
```

`RTC(R)` specifies the reflexive transitive closure of rewriting relation `R`.

The lemma `Linear_and_Non_ambiguous_implies_confluent` is proved in a standard manner being much simpler than the last one on confluence of orthogonal systems. In fact, since, in addition to orthogonality restrictions, variables cannot appear repeatedly in the right-hand side of the rules this proof does not need elaborated manipulation of reductions and instantiations in order to build the term of parallel joinability for divergence terms.

By the specification of these lemmas, one can observe that `Orthogonal_implies_confluent`, that is the main lemma, depends on the formalization of `parallel_reduction` and `parallel_reduction_is_DP` . The latter lemma is relatively simple and the former is the crucial one.

In order to classify overlaps in a parallel divergence from a term in which, on the one side, a parallel rewriting is applied at positions Π_1 and, on the other side, at positions Π_2, positions involved in a parallel divergence are classified through the following specified recursive relations:

```
sub_pos((fsp : PP), p : position): RECURSIVE finseq[position] =
     IF  length(fsp) = 0 THEN empty_seq[position]
        ELSIF p <= fsp(0) AND p /= fsp(0)
                THEN add_first(fsp(0), sub_pos(rest(fsp), p))
                ELSE sub_pos(rest(fsp), p)
     ENDIF
    MEASURE length(fsp)

Pos_Over((fsp1 : PP), (fsp2 : PP)): RECURSIVE finseq[position] =
   (IF length(fsp1) = 0
       THEN empty_seq[position]
       ELSE (IF ( length(sub_pos(fsp2, fsp1(0))) > 0
                 OR PP?(add_first(fsp1(0), fsp2)))
                THEN add_first(fsp1(0), Pos_Over(rest(fsp1), fsp2))
                ELSE Pos_Over(rest(fsp1), fsp2)
             ENDIF)
     ENDIF)
    MEASURE length(fsp1)
```

`sub_pos`(Π, π) builds the subsequence of positions of the sequence of parallel positions Π that are strictly below the position π; that is, $\pi' \in \Pi$ such that π is a prefix of π', as usual denoted as $\pi < \pi'$.

`Pos_Over`(Π_1, Π_2) builds the subsequence of positions from Π_1 that are parallel to all positions in Π_2 or that have positions in the sequence Π_2 below them. In this specification, `PP?` is a predicate for the type `PP` of sequences of parallel positions.

As will be explained, these functions are crucial in order to build the term of one-step parallel joinability, necessary in the proof of lemma `parallel_reduction_is_DP`.

4 Confluence of Orthogonal TRSs

Here, an analytic proof of the main theorem is presented. The notational conventions, colors and figures that are used in this proof are followed in order to guide the explanation of the formalization presented in the next section.

Theorem 4.1 (`Orthogonal_implies_confluent`) *An orthogonal TRS is confluent.*

This theorem is proved according to the following sketch:

- firstly, constructing the relation of *parallel reduction* $\rightrightarrows$ associated with the rewriting relation $\rightarrow$, as it was defined.
- Afterwards, it is proved that
$$\rightarrow \subseteq \rightrightarrows \subseteq \rightarrow^*$$
from which one concludes that $\rightrightarrows^* = \rightarrow^*$. The lemma `parallel_reduction`, whose specification was presented at the end of the previous section, corresponds to the latter inclusion.
- Finally, it is proved that for orthogonal systems, $\rightrightarrows$ has the diamond property, which corresponds to the lemma `parallel_reduction_is_DP` presented in the previous section. So one obtains confluence of $\rightarrow$:

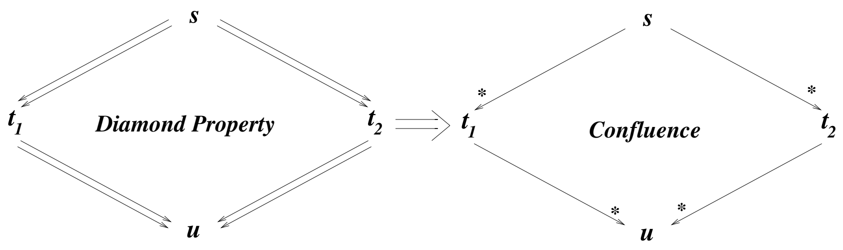

Figure 3: Diamond property of parallel rewriting implies confluence

Theorem 4.2 (Orthogonality implies diamond property) *Let R be um TRS orthogonal. Then, the relation $\rightrightarrows$ has the diamond property.*

Proof. Let $s, t_1, t_2 \in T(\Sigma, V)$ such that $t_1 \leftleftarrows s \rightrightarrows t_2$. Then, there are two sequences Π_1 and Π_2 of parallel positions of s where reductions associated with the parallel reductions $t_1 \leftleftarrows s$ and $s \rightrightarrows t_2$ happen. Let ε_1 and Γ_1 denote respectively the sequences of rules and substitutions associated with the parallel reduction $t_1 \leftleftarrows s$ and ε_2 and Γ_2 the sequences of rules and substitutions associated with the parallel reduction $s \rightrightarrows t_2$. The situation is sketched in Fig. 4. Subterms at positions Π_1 and Π_2 are respectively represented as red and blue triangles, except for subterms at positions in the intersection that are represented as purple triangles. As illustrated in the figure, some subterms at poisitions Π_1 are inside subterms at positions Π_2 and vice versa. This is what makes elaborated the necessary analysis in order to obtain the proof.

A first consequence of non-ambiguity is that purple subterms are reduced in both parallel reductions in the same manner. Thus they are trivially joined. Another consequence of non-ambiguity is that all red subterms occurring inside a blue subterm, occur either at or below a variable position of the left-side of the blue rule being applied in the parallel reduction $s \rightrightarrows t_2$ and vice versa. In other words, if $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, are positions such that π_1 inside π_2, that is, there exists π such that $\pi_1 = \pi_2\pi$, then, for the rule $l \rightarrow r$ and substitution σ associated with position π_2 in the sequences ε_2 and Γ_2, respectively, there exist π' and π'' such that $\pi'\pi'' = \pi$ and $l|_{\pi'}$ is a variable, said x. One observes that $x\sigma|_{\pi''}$ is the red subtem of s at position π_1. Notice that when π'' is the root position, the subterm occurs exactly at a variable position of the left-hand side of the rule. Also, notice that otherwise, when the red subterm is supposed to occur at a non variable position, this contradicts the assumption of being non-ambiguous. To emphasize this, in figure 4 all red subterms below blue subterms and vice versa were included inside a black pentagon.

Figure 5 illustrates the parallel divergence. On the one side, red triangles are reduced into painted red triangles; on the other side, blue triangles are reduced into painted blue triangles. Since in orthogonal

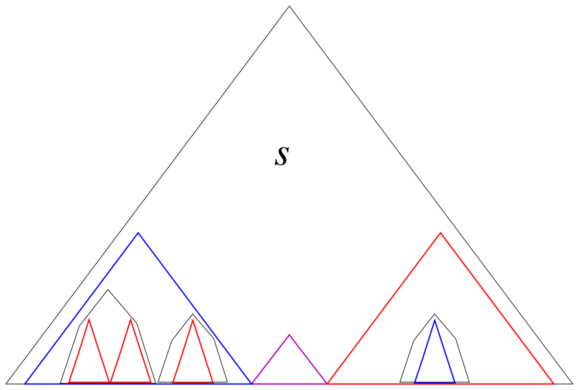

Figure 4: Subterms at positions Π_1 and Π_2

systems right linearity is not obligatory, rewriting rules can repeat variables to the right. This is illustrated in the figure as follows: in the term t_1, the subterm illustrated by the red triangle to the right reduces through a rule that repeats twice the variable in whose instantiation a blue subterm occurs; in the term t_2 the subterm represented by the blue triangle to the left reduces through a rule that repeats twice a variable in whose instantiation two red subterms occur.

To prove the diamond property, a term u should be built that is reached in one parallel step both from t_1 and t_2 as illustrated in the figure 6.

In the sequel, the parallel rewriting $t_1 \rightrightarrows_E u$ will be justified. Understanding how this is possible for t_1, one will understand how this symmetrically happens for $t_2 \rightrightarrows_E u$ as well. Basically, three cases should be considered.

1. As previously observed, for purple subterms, that are those at positions $\pi \in \Pi_1 \cap \Pi_2$, the rules applied to the left and to the right are the same; that is, for $\pi \in \Pi_1 \cap \Pi_2$, $t_1|_\pi \leftarrow s|_\pi \rightarrow t_2|_\pi$ and $t_1|_\pi = t_2|_\pi$. This coincidence is represented in the figures 6 and 7 as a painted purpled triangle.
2. For blue subterms inside red subterms as illustrated by the largest red triangle in figure 7, suppose the position of the subterm and the associated rule and substitution are $\pi \in \Pi_1, g \rightarrow d, and \sigma$, respectively. Let $\Pi_\pi = \{\pi_1, \dots \pi_k\} \subseteq \Pi_2$ be the subset of all positions of Π_2 below π. For $1 \leq j \leq k$, let $l_j \rightarrow r_j$ and σ_j denote the rule and substitution associated with position π_j. Then, according to a previous observation, for all $1 \leq j \leq k$, there exist π'_j and π''_j such that $\pi\pi'_j\pi''_j = \pi_j$, being π'_j a variable position of the left-hand side of the rule $g \rightarrow d$. Let σ' the substitution obtained from σ modifying all variables according to substitutions σ_j, then, the divergence at position π, that is $t_1|_\pi \leftleftarrows s|_\pi \rightrightarrows t_2|_\pi$ (see transformation of the biggest red triangle in figure 6) can be joined in one step of parallel reduction as $t_1|_\pi = d\sigma \rightrightarrows d\sigma' \leftarrow g\sigma' = t_2|_\pi$. The construction of the substitution σ' is one of the most elaborated steps in the formalization of this theorem. Namely, suppose x is a variable occurring in the left-hand side of the rule $g \rightarrow d$ only at position π' (left-linearity guarantees unicity of π'); if $\pi' \neq \pi'_j$, for all $1 \leq j \leq k$, then $x\sigma' := x\sigma$. Otherwise, let $\{j_1, \dots j_m\}$ be the set of indices such that $\pi' = \pi'_{j_l}$, for $1 \leq l \leq m$ and $1 \leq j_l \leq k$. Since Π_π are parallel positions, $\{\pi''_{j_1}, \dots \pi''_{j_m}\}$ are parallel positions of $x\sigma$. By applying the rules $l_{j_l} \rightarrow r_{j_l}$ with substitutions σ_{j_l}, for $1 \leq l \leq m$, one reduces in parallel $x\sigma \rightrightarrows x\sigma[\pi''_{j_1} \leftarrow r_{j_1}\sigma_{j_1}] \dots [\pi''_{j_m} \leftarrow r_{j_m}\sigma_{j_m}]$. Thus, in this case, $x\sigma'$ is defined as $x\sigma[\pi''_{j_1} \leftarrow r_{j_1}\sigma_{j_1}] \dots [\pi''_{j_m} \leftarrow r_{j_m}\sigma_{j_m}]$.
3. For red subterms inside blue subterms the construction is symmetric to the one of the previous

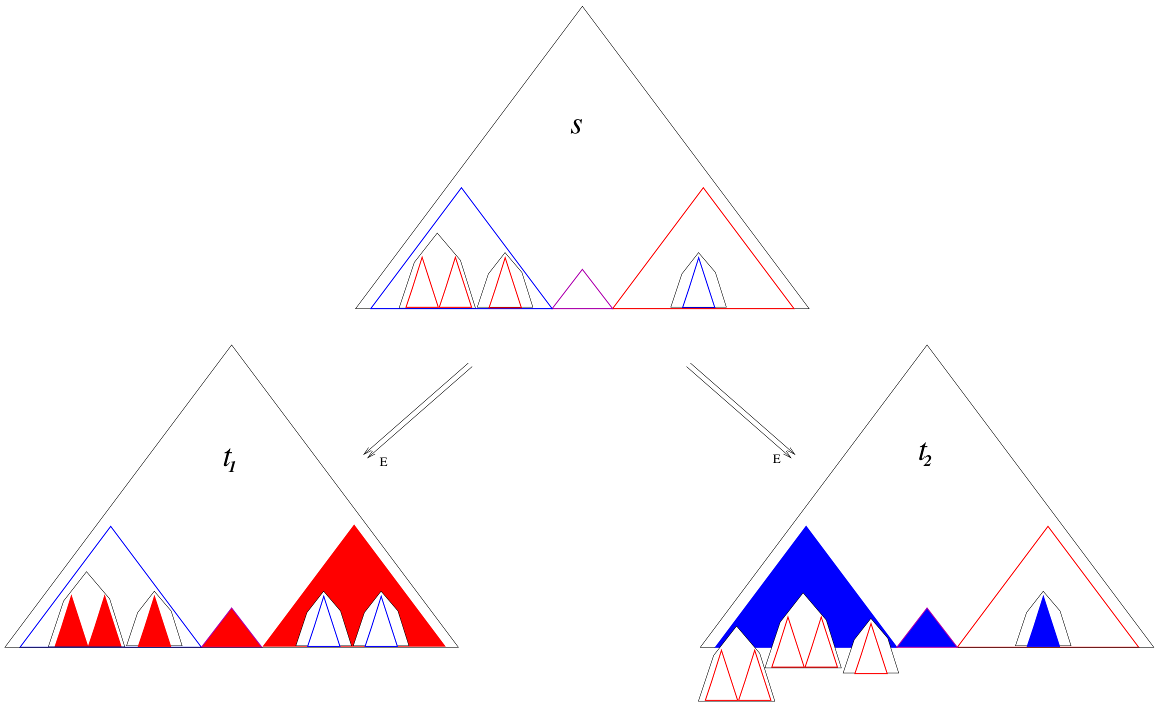

Figure 5: One-step parallel divergence

item, as illustrated by the biggest blue triangle in figure 7. Suppose the position of this subterm and the associated rule and substitution are $\pi \in \Pi_2, l \to r, and \sigma$, respectively. Then a substitution σ' can be built as before, such that the divergence at position π, that is $t_1|_\pi \leftleftarrows s|_\pi \rightrightarrows t_2|_\pi$ (see transformation of the biggest blue triangle in figure 6) can be joined in one step of parallel reduction as $t_1|_\pi = l\sigma' \to r\sigma' \leftleftarrows r\sigma = t_2|_\pi$.

5 Formalization of Confluence of Non Ambiguous and Linear TRSs

Computational formalizations do not admit mistakes and, in particular, those specifications based on rewriting rules as well as on recursive functional definitions can profit from a formalization of confluence of orthogonality. Several works report efforts on specification of different computational objects (software and hardware) through term rewriting systems (e.g., [1, 2, 9, 10]) . Consequently, it is relevant to have robust and as complete as possible libraries for the theory of abstract reduction and term rewriting system in different proof assistants.

As mentioned in the previous section, the adopted proof strategy is to construct the joinability term for the Theorem 4.2 (that is lemma `parallel_reduction_is_DP` , in the formalization, term u in Fig. 6) for the parallel divergence. For doing this, it's necessary to prove that there are positions, rules and substitutions sequences that satisfy the requirements.

Following the approach given in the analytic proof of Theorem 4.2, one of the divergence terms is chosen and one built the sequences. Consider for instance the term t_1 of the divergence in Fig. 7 (see also the cases in the three items in the proof of this theorem). One has to rewrite the blue subterms to get the joinability term u.

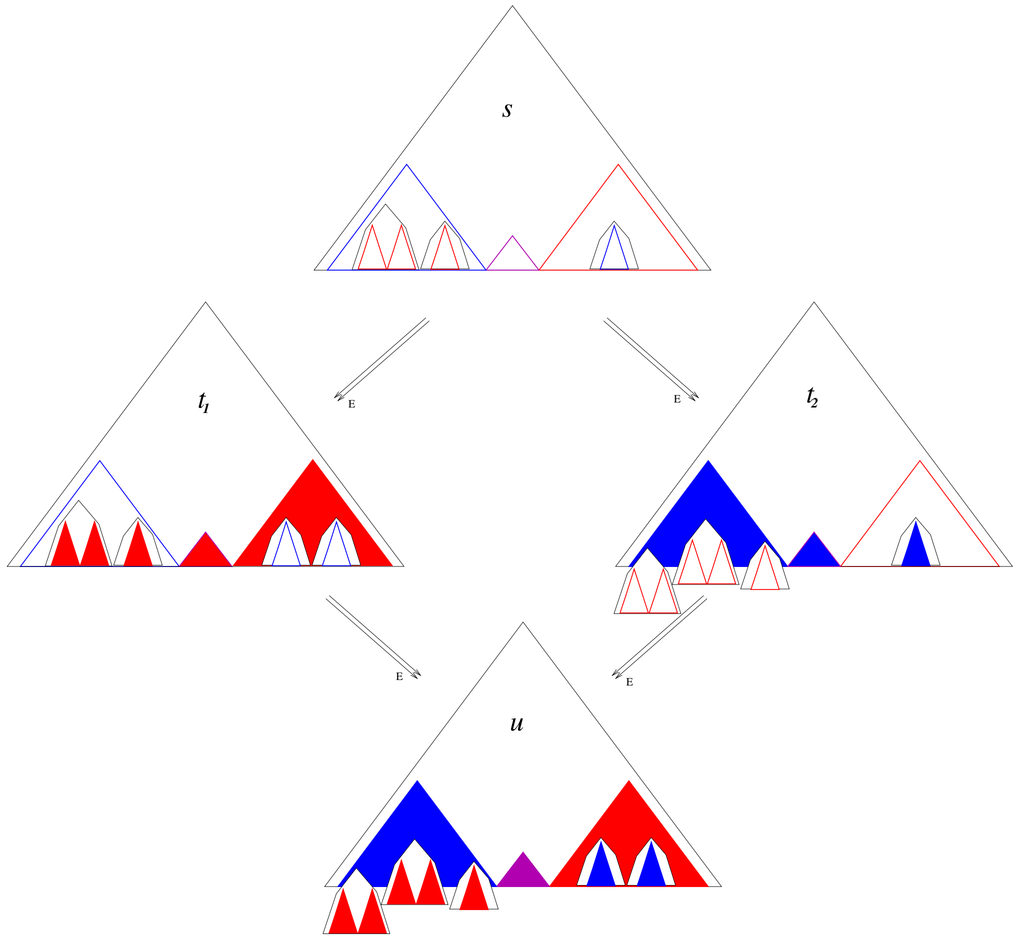

Figure 6: Diamond property of parallel rewriting

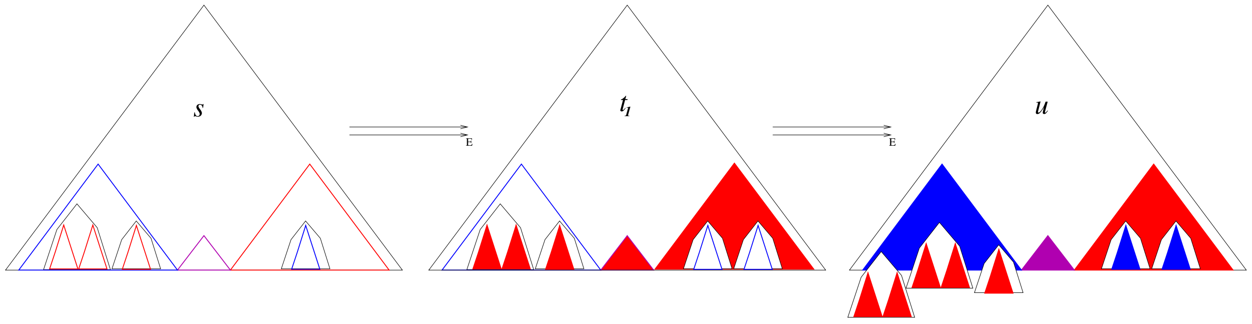

Figure 7: One-step parallel rewriting from t_1 into u

As in the analytical proof, one has to consider the cases that will be presented in the next subsections. The colors convention applied in the explanation of this theorem is used.

5.1 Rewriting blue suberms above red painted ones

Remember that Pos_Over(Π_2, Π_1) builds the subsequence of positions from Π_2 that are parallel to all positions in Π_1 or that have positions in the sequence Π_1 below them. For $\pi \in \Pi_2$, sub_pos(Π_1, π) builds the subsequence Π_π of positions in Π_1 below π.

In order to construct the subsequence of either rules or substitutions associated with a subsequence of positions Π' of Π, from Γ and Σ, respectively (see the definition of parallel reduction), the function choose_seq was specified. Using this function one can build the subsequences of rules and substitutions associated with the positions Π_π, for instance, calling choose_seq$(\Pi_\pi, \Pi_1, \Gamma_1)$ or choose_seq$(\Pi_\pi, \Pi_1, \Sigma_1)$, respectively. choose_seq is a polymorphic function that can be used for several tasks. In particular, in the construction of the substitution σ', it can be used in order to choose the sequence of terms, instantiations of rhs of rules, that should be changed in order to obtain $x\sigma'$, for a variable x occurring at position $\pi\pi'$; namely, this is done calling choose_seq(sub_pos$(\Pi_1, \pi\pi')$, $\Pi_1, \{d_1\sigma_1, \ldots, d_n\sigma_n\}$), where the sequence of terms $\{d_1\sigma_1, \ldots, d_n\sigma_n\}$ is straightforwardly built from the sequences of rules and substitutions associated with Π_1, i.e., $\Gamma_1 = \{g_1 \to d_1, \ldots, g_n \to d_n\}$ and $\Sigma_1 = \{\sigma_1, \ldots, \sigma_n\}$.

```
  choose_seq(seq:PP, seq1:PP, (seq2 | seq1`length=seq2`length)):
  RECURSIVE finseq[T] =
      IF length(seq1)=0 THEN empty_seq
         ELSIF mem_seq(seq1(0),seq)
            THEN add_first(seq2(0),choose_seq(seq,rest(seq1),rest(seq2)))
            ELSE choose_seq(seq,rest(seq1),rest(seq2))
      ENDIF
      MEASURE(length(seq1))
```

The construction of the substitution σ', explained in Theorem 4.2, requires the specification of two recursive functions SIGMA and SIGMAP.

```
SIGMA(sigma, x, fst, (fsp:SPP(sigma(x))|length(fsp)=length(fst)))(y:(V)):
     term = IF length(fst)=0 OR y/=x
            THEN sigma(y)
            ELSE replace_terms(sigma(x),fst,fsp)
         ENDIF
```

SIGMA has as arguments the original substitution σ, a comparison variable x and the associated subsequences of substituting terms and positions relative to the necessary update of $x\sigma$. In the notation applied in the proof of Theorem 4.2, SIGMA$(\sigma, x, \{d_{j_1}\sigma_{j_1}, \ldots, d_{j_m}\sigma_{j_m}\}, \{\pi''_{j_1}, \ldots, \pi''_{j_m}\})$ applied to x will give $x\sigma'$, that is $x\sigma[\pi''_{j_1} \leftarrow d_{j_1}\sigma_{j_1}] \ldots [\pi''_{j_m} \leftarrow r_{j_m}\sigma_{j_m}]$.

The construction of the whole substitution σ', is done through the function SIGMAP below, that adequately calls the function SIGMA. SIGMAP$(\sigma, \{x_1, \ldots, x_q\}, \{\pi\pi'_1, \ldots, \pi\pi'_q\}, \{d_1\sigma_1, \ldots, d_n\sigma_n\}, \{\pi_1, \ldots, \pi_n\})$, where $\{x_1, \ldots, x_q\}$ and $\{\pi\pi'_1, \ldots, \pi\pi'_q\}$ are the sequence of variables at lhs of the rule $l \to r$ that should change, assuming $l\sigma$ occurs at position π, and the associated sequence of positions of these variables in the whole term t_1, respectively. For a variable $y \in \{x_1, \ldots, x_q\}$, say $y = x_r$, SIGMAP calls the function SIGMA giving as input the sequence of terms to be substituted and their associated positions in $y\sigma$. This is done through application of the functions choose_seq and complement_pos. The former one, is called as choose_seq($\{\pi\pi_r'\pi_{r,j_1}'', \ldots, \pi\pi_r'\pi_{r,j_{m_r}}''\}, \{\pi_1, \ldots, \pi_n\}, \{d_1\sigma_1, \ldots, d_n\sigma_n\}$), which gives the sequence of substituting terms. The latter one is called as complement_pos($\pi\pi_r', \{\pi_1, \ldots, \pi_n\}$), which gives as result the associated positions inside $l\sigma$, that is $\{\pi_{r,j_1}'', \ldots, \pi_{r,j_{m_r}}''\}$.

```
SIGMAP(sigma,fsv,(fsp1:PP|fsp1`length=fsv`length),
       fst,(fsp2:PP|fsp2`length=fst`length))(y:(V)):
     RECURSIVE term=
        IF length(fsv)=0 THEN sigma(y)
           ELSIF y=fsv`seq(0) & SP?(sigma(fsv`seq(0)))(complement_pos(fsp1`seq(0),fsp2))
              THEN SIGMA(sigma,fsv`seq(0),choose_seq(sub_pos(fsp2,fsp1`seq(0)),fsp2,fst),
                          complement_pos(fsp1`seq(0),fsp2))(y)
              ELSE SIGMAP(sigma,rest(fsv),rest(fsp1),fst,fsp2)(y)
        ENDIF
     MEASURE(length(fsv))
```

A few number of lemmas were formalized in order to prove soundness of this definition. Namely, the fact that it is in fact a substitution is axiomatized. Among these lemmas, as a matter of illustration, it is necessary to prove that the subsequences of terms and positions given as third and second parameters of the call of SIGMA have the same length. This is stated as the following lemma easily formalized by induction on the length of the finite sequences.

```
complement_pos_preserv_sub_pos_length1: LEMMA
  PP?(fsp) IMPLIES
 complement_pos(p, fsp)`length = sub_pos(fsp, p)`length
```

5.2 Rewriting blue subterms inside red painted ones

This is the case in which an instance of $g\sigma$ occurring at position $\pi \in \Pi_1$ reduces into $d\sigma$ (see the big red triangle in Figs. 6 and 7). In this case, one has to show that $d\sigma \rightrightarrows d\sigma'$.

Among the necessary functions to specify this parallel reduction, one has compo_pos_var listed below that specifies the construction of the sequence of all positions inside $d\sigma$ that should be rewritten in order to obtain $d\sigma'$. The call compo_pos_var($g, d, \{x_1, \ldots, x_q\}, \pi, \Pi_2$), where $\pi \in \Pi_1$ is the position in which $g\sigma$ occurs in t_1, builds the finite sequence of positions in $d\sigma$ that should be rewritten, that is, the blue triangles inside pentagons in the big red painted triangle in figure 7. Recursively, for each variable $x_j \in \{x_1, \ldots, x_q\}$, the following is done:

- Initially, choose(Pos_var(g, x_j)) builds the position π_j' in which x_j appears in g and through complement_pos(π, Π_2) one builds the sequence of positions $\{\pi_j'\pi_{j_1}'', \ldots, \pi_j'\pi_{j_k}''\}$ of $g\sigma$ according to the notation in the proof of Theorem 4.2.
- Afterwards, applying complement_pos to the previous results one obtains the sequence of positions $\{\pi_{j_1}'', \ldots, \pi_{j_k}''\}$. Also, set2seq(Pos_var($d, x_j$)) computes the sequence of positions in which x_j occurs in d, denoted as $\{\gamma_1, \ldots \gamma_r\}$.
- Finally, compo_pos_multi($\{\gamma_1, \ldots \gamma_r\}, \{\pi_{j_1}'', \ldots, \pi_{j_k}''\}$) builds the necessary sequence of all positions in d that should be modified, that is $\{\gamma_1\pi_{j_1}'', \ldots, \gamma_1\pi_{j_k}'', \ldots, \gamma_r\pi_{j_1}'', \ldots, \gamma_r\pi_{j_k}''\}$ and compo_pos concatenates the prefix π to this sequence.

```
compo_pos_var(g,d,fsv,p,(fsp:PP)): RECURSIVE finseq[position] =
          IF length(fsv)=0 THEN empty_seq
             ELSIF nonempty?(Pos_var(g,fsv`seq(0)))
                  THEN compo_pos(p,compo_pos_multi(set2seq(Pos_var(d,fsv`seq(0))),
                                                   complement_pos(choose(Pos_var(g,fsv`seq(0))),
                                                                  complement_pos(p,fsp))))
                        o  compo_pos_var(g,d,rest(fsv),p,fsp)
                  ELSE compo_pos_var(g,d,rest(fsv),p,fsp)
          ENDIF
      MEASURE(length(fsv))
```

Analogously to the previous function other ones as compo_rr_var and compo_Sub_var build the associated sequences of blue rewrite rules and corresponding blue substitutions used to replace subterms at positions $\{\gamma_1\pi''_{j_1},\ldots,\gamma_1\pi''_{j_k},\ldots,\gamma_r\pi''_{j_1},\ldots,\gamma_r\pi''_{j_k}\}$ in order to obtain $d\sigma'$.

As previously mentioned a few technical axiomatizations are included in the whole PVS development, that need to be fully formalized in order to have a complete formalization of the theorem of confluence of orthogonal TRSs. Among these axiomatized properties, in relation with Compo_pos_var, it is necessary to formalize the fact that the computed sequence $\{\pi\gamma_1\pi''_{j_1},\ldots,\pi\gamma_1\pi''_{j_k},\ldots,\pi\gamma_r\pi''_{j_1},\ldots,\pi\gamma_r\pi''_{j_k}\}$ is a sequence of parallel positions. Preservation of the ordering and consequently of the association of components in the sequences of positions, substitutions, rules, substituting terms, etc. computed with functions as choose_seq, compo_pos_var, etc. is a consequence of the conservative recursive way in that these sequences were built and all the necessary related lemmas were formalized.

Currently, the whole development consist of among 1.300 lines of specification and 46.000 lines of proofs. Indeed, there are 40 definitions, 84 proved lemmas and 8 axioms.

6 Related work and Conclusions

PVS specifications of non trivial notions and formalizations of results of the theory of rewriting were presented, that are related with the properties of parallel rewriting and orthogonal rewriting systems. The PVS theory for orthogonal TRSs enriches the PVS theory trs for TRSs introduced in [5] and available in [12]. The formalization of these properties of orthogonal TRSs are close to the analytical proofs presented in textbooks as [3] and [4], which provides additional evidence of the appropriateness of both the higher-order specification language and the proof engine of PVS to deal in a natural way with specification of rewriting notions and properties and their formalizations. This consequently implies the good support of PVS to deal with soundness and completeness and integrity constraints of specifications of computational objects specified through rewriting rules.

In its current status, the theory for orthogonal TRSs includes a complete formalization of confluence of non-ambiguous and linear TRSs as well as a proof of confluence of orthogonal TRSs which depends on both the lemma of equivalence of the reflexive-transitive closure of the rewriting and the parallel reduction relations and of the lemma of diamond property of the parallel reduction relation of orthogonal TRSs. The latter lemma is formalized axiomatizing some technical properties of parallel positions, rules and substitutions involved in one-step of parallel reduction. In [11] the criterion of weak orthogonality was integrated to ensure confluence applying the certification tool CeTA. Instead orthogonality, weak orthogonality allows for trivial critical pairs. To the best of our knowledge no formalization of the property of confluence of orthogonal TRSs is available in any proof assistant.

References

[1] Arvind & X. Shen (1999): *Using Term Rewriting Systems to Design and Verify Processors*. *IEEE Micro* 19(3), pp. 36–46.

[2] M. Ayala-Rincón, C. Llanos, R. P. Jacobi & R. W. Hartenstein (2006): *Prototyping Time and Space Efficient Computations of Algebraic Operations over Dynamically Reconfigurable Systems Modeled by Rewriting-Logic*. *ACM Transactions on Design Automation of Electronic Systems* 11(2), pp. 251–281.

[3] F. Baader & T. Nipkow (1998): *Term Rewriting and* All That. Cambridge University Press.

[4] M. Bezem, J.W. Klop & R. de Vrijer, editors (2003): *Term Rewriting Systems by TeReSe*. *Cambridge Tracts in Theoretical Computer Science* 55, Cambridge University Press.

[5] A. L. Galdino & M. Ayala-Rincón (2009): *A PVS* Theory *for Term Rewriting Systems*. In: *Proceedings of the Third Workshop on Logical and Semantic Frameworks, with Applications - LSFA 2008*, *Electronic Notes in Theoretical Computer Science* 247, pp. 67–83.

[6] A. L. Galdino & M. Ayala-Rincón (2010): *A Formalization of the Knuth-Bendix(-Huet) Critical Pair Theorem*. *J. of Automated Reasoning* 45(3), pp. 301–325.

[7] G. Huet (1980): *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*. *Journal of the Association for Computing Machinery* 27(4), pp. 797–821.

[8] D. E. Knuth & P. B. Bendix (1970): *Computational Problems in Abstract Algebra*, chapter Simple Words Problems in Universal Algebras, pp. 263–297. J. Leech, ed. Pergamon Press, Oxford, U. K.

[9] C. Morra, J. Becker, M. Ayala-Rincón & R. W. Hartenstein (2005): *FELIX: Using Rewriting-Logic for Generating Functionally Equivalent Implementations*. In: *15th Int. Conference on Field Programmable Logic and Applications - FPL 2005*, IEEE CS, pp. 25–30.

[10] C. Morra, J. Bispo, J.M.P. Cardoso & J. Becker (2008): *Combining Rewriting-Logic, Architecture Generation, and Simulation to Exploit Coarse-Grained Reconfigurable Architectures*. In Kenneth L. Pocek & Duncan A. Buell, editors: *FCCM*, IEEE Computer Society, pp. 320–321. Available at `http://dx.doi.org/10.1109/FCCM.2008.37`.

[11] R. Thiemann (2012): *Certification of Confluence Proofs using CeTA*. In: *First International Workshop on Confluence(IWC 2012)*, p. 45.

[12] Theory `trs` (consulted January 2012): *Available in the NASA LaRC PVS library,* `http://shemesh.larc.nasa.gov/fm/ftp/larc/ PVS-library/pvslib.html`.