Formalizing the Confluence of Orthogonal Rewriting Systems*

Ana Cristina Rocha Oliveira† and Mauricio Ayala-Rincón‡
Grupo de Teoria da Computação
Departamentos de Matemática e Ciência da Computação
Universidade de Brasília
Brasília D.F., Brazil
Email: anacrismarie@gmail.com, 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 should coincide. Although the proofs are 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 has been formalized, based on axiomatizations of properties of rules, positions and substitutions involved in parallel steps of reduction, in this proof assistant. Proofs for some similar but restricted properties such as the property of confluence of non-ambiguous and (left and right) linear TRSs have been fully formalized.

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, 9], which establishes that whenever all critical pairs of a given terminating rewriting system are joinable, the system is confluent. This criterion as well as other criteria for abstract reduction systems such as Newman's lemma were fully formalized in the proof assistant PVS in [5, 7] over the PVS *theory* `trs` [6], that is available in the NASA LaRC PVS library [14]. 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* and *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.

*Work supported by grants from CNPq/Universal, CAPES/STIC-AmSud and FAPDF/PRONEX.
†Author supported by CAPES.
‡Author partially supported by CNPq.

D. Kesner and P. Viana (Eds.): LSFA 2012
EPTCS 113, 2013, pp. 145–152, doi:10.4204/EPTCS.113.14

This work reports 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 relation were included in order to follow the standard inductive proof approach of this property, that is based on the proof of the diamond property for the parallel reduction associated to any orthogonal TRS as presented in [3]. 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 TRSs. An extended version of this paper as well as the PVS development are available at `www.mat.unb.br/~ayala/publications.html`.

Proofs of confluence of orthogonal TRSs have been known at least since Rosen's seminal study on Church-Rosser properties of *tree manipulating systems* [12] and several of them are based on a similar strategy through the famous Parallel Moves Lemma. Rosen's proof uses a notion of residuals of positions in a notation that was standardized further by Huet in [8], paper in which Huet presented a proof of confluence of left-linear and parallel closed TRSs that unlike Rosen's proof admits critical pairs that should be joinable from left to right in a sole step of parallel reduction.

In the chapter on orthogonality of [4], the authors presented five styles of proof of confluence of orthogonal systems as well as an extension of the confluence result to weakly orthogonal TRSs. All the given styles of proof are not different in essence: the first one uses the notions of residuals and descendants via the parallel moves lemma, the second one avoids explicit mention of residuals by underlining reductions, the third one imports from λ-calculus and combinatory logic the notion of complete developments and the fourth style uses elementary and reduction diagrams. The fifth given proof is an inductive confluence proof that is the more related with our approach of formalization and follows lines of reasoning based on analysis of properties of the parallel rewriting relation and the parallel moves lemma, just by changing the definition of parallel reduction. In this proof the parallel relation is defined from the rewriting relation as the reflexive relation that is compatible with substitutions and, parallely compatible with contexts. Thus, after proving a version of the parallel moves lemma, the diamond property of the parallel reduction is proved by induction on the structure of terms based on the analysis of the six possible cases of a parallel divergence; that is, whether the divergence terms are obtained from a term by application of two different steps of parallel reduction, by combinations of reflexivity, substitution and context according to the definition of the parallel relation. In this analysis, the version of the parallel moves lemma is applied for the case of a divergence in which on the one side a term is obtained by substitution and on the other side by context.

For this formalization it has been chosen the inductive proof presented in [3] because it uses the nowadays standard rewriting notation as the PVS *theory* `trs` does, uses a standard definition of parallel reduction and follows lines of reasoning that from the authors' viewpoint are of great didactical interest.

2 Specification of basic Notions and Definitions

Standard notation of of the theory of rewriting is used as in [3] or [4]. One says that a rewriting relation $\rightarrow$:

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

A well-defined set of terms is built from a given signature and an enumerable set of variables. A rule $e = (l, r)$ is an ordered pair of terms such that the first one cannot be a variable and all variables in the

second one occur in the first one. A TRS is given as a set of rules. The reduction relation $\rightarrow_E$ induced by a TRS E is built as follows: a term t reduces to t_0 (denoted as $t \rightarrow t_0$) if there are a position π of 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 by the corresponding instantiation of the right-hand side (**rhs**, for short) of the rule, that is $rhs(e)\sigma$. The only change done in order to obtain t_0 from t, occurs at position π. All this is summarized by the following notation: $t = t[\pi \leftarrow lhs(e)\sigma] \rightarrow_E t[\pi \leftarrow rhs(e)\sigma] = t_0$, 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.

Given terms t_1 and t_2, one says that t_1 reduces in parallel to t_2, denoted as $t_1 \rightrightarrows_E t_2$, whenever there exist finite sequences

$$\Pi := \pi_1, \ldots, \pi_n;$$

$$\Sigma := \sigma_1, \ldots, \sigma_n \text{ and}$$

$$\Gamma := e_1, \ldots, e_n$$

of parallel positions of t_1, substitutions and rules in E, respectively, such that:

$$t_1|_{\pi_i} = lhs(e_i)\sigma_i, \forall i = 1, \ldots, n,$$

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

$$t_2|_{\pi_i} = rhs(e_i)\sigma_i, \forall i = 1, \ldots, n,$$

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:

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

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

The PVS *theory* `trs` includes all necessary basic notions and properties to formalize elaborated theorems of the theory of rewriting such as the one of confluence of orthogonal systems. `trs` includes specifications and formalizations of the algebra of terms, subterms and positions, properties of abstract reduction systems, confluence and termination, among others. The current development of the PVS *theory* called `orthogonality` deals specifically with orthogonality related notions and properties. Among the definitions specified inside the *theory* `orthogonality` one could mention the basic boolean ones listed below, where `E` is a set of rewriting rules (equations).

```
- 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) & Right_Linear?(E)

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

In the specification of Ambiguous?(E), CP?(E)(t1,t2) specifies that t1 and t2 are critical pairs of the rewriting system E. A term t is linear? whenever, each variable x in t occurs only once. The expressions Right_Linear?(E) and Left_Linear?(E) indicate respectively that the rhs and the lhs of all rules in E are linear. The predicate Linear? specifies linearity of sets of rewriting rules. Finally, Orthogonal? specifies orthogonality of TRSs.

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(i), E)), fss) :
      fsp'length = fse'length & fsp'length = fss'length
      & (FORALL (i : below[fsp'length]) : subtermOF(s, fsp(i)) = ext(fss(i))(lhs(fse(i))))
      & t = replace_par_pos(s, fsp, fse, fss)
```

RC(R), that is used in local_joinability_triangle?, specifies the reflexive closure of the rewriting relation R. For a functional term s, f(s) and args(s) compute the head function symbol of s and its arguments respectively; also, app(f(s), args(s)) builds the functional term s. The function replace with arguments (t, st, i) replaces the $(i+1)^{th}$ term of the sequence of arguments st by t. The recursive function replaceTerm replaces a subterm of a term: it gives as output for the input triplet (s, t, p) the term obtained from s by replacing the subterm at position p of s by t, that in standard rewriting notation is written as $s[p \leftarrow t]$. Similarly, replace_par_pos specifies the parallel replacements necessary in one step of parallel reduction. The specification of the relation of parallel reduction is given by parallel_reduction?, in which the variables fsp, fse and fss are the sequences of parallel positions, rewrite rules and substitutions, that were denoted respectively as Π, Γ and Σ, in the definition of the parallel reduction relation.

The main lemmas and theorems specified and formalized 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) & NOT Ambiguous?(E) => local_joinability_triangle?(reduction?(E))

- One_side_diamond_implies_conflent: LEMMA local_joinability_triangle?(R) => confluent?(R)

- Linear_and_Non_ambiguous_implies_confluent: LEMMA
```

```
  FORALL (E) : ((Linear?(E) & NOT Ambiguous?(E) ) => 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), that is used in parallel_reduction, specifies the reflexive transitive closure of the rewriting relation R.

The lemma Linear_and_Non_ambiguous_implies_confluent is proved in a standard manner. In fact, since, in addition to orthogonality restrictions, variables cannot appear repeatedly in the rhs 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 former lemma is relatively simple and the latter is the crucial one.

In order to classify overlaps in a parallel divergence from a term in which, on the one side, a parallel reduction 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) & 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. 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.

Confluence of orthogonal TRSs is proved according to the following sketch: Firstly, it is proved $\rightarrow \subseteq \rightrightarrows \subseteq \rightarrow^*$, from which one concludes that $\rightrightarrows^* = \rightarrow^*$. The lemma parallel_reduction corresponds to the latter inclusion. Then, it is proved that for orthogonal systems, $\rightrightarrows$ has the diamond property, which corresponds to the lemma parallel_reduction_is_DP. For an analytical proof see the extended version of this paper.

3 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 TRSs (e.g., [1, 2, 10, 11]). Consequently, it is relevant to have robust and as complete as possible libraries for the theory of abstract reduction systems and TRSs in different proof assistants. To construct the joinability term for the lemma `parallel_reduction_is_DP`, one has to consider several cases from which one is explained in the sequel.

Suppose, one has a parallel divergence from term $t_1 \leftleftarrows s \rightrightarrows t_2$ at positions Π_1 and Π_2 with respective associated rules and substitutions Γ_i and Σ_i, $i \in \{1,2\}$. Let $\pi \in \Pi_2$ and $l \to r$ and σ denote the associated rule and substitution, respectively. $\texttt{sub\_pos}(\Pi_1, \pi)$ builds the subsequence Π_π of positions in Π_1 below π. Let $\Pi_\pi = \{\pi_1, \ldots \pi_k\}$. For $1 \leq j \leq k$, let $g_j \to d_j$ and σ_j denote the rule and substitution associated with position π_j. Then, by non-ambiguity, 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 lhs of the rule $l \to r$.

Let σ' be the substitution obtained from σ modifying all variables according to substitutions σ_j, then, the divergence at position π, that is $t_2|_\pi \leftleftarrows s|_\pi \rightrightarrows t_1|_\pi$ can be joined in one step of parallel reduction as $t_2|_\pi = r\sigma \rightrightarrows r\sigma' \leftarrow l\sigma' = t_1|_\pi$. The construction of σ' is one of the most elaborated steps in this formalization. Namely, suppose x is a variable occurring in the lhs of the rule $l \to r$ 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 $\{j_1, \ldots, j_m\}$ 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, $\{\pi''_{j_1}, \ldots \pi''_{j_m}\}$ are parallel positions of $x\sigma$. By applying the rules $g_{j_l} \to d_{j_l}$ with substitutions σ_{j_l}, for $1 \leq l \leq m$, one reduces in parallel $x\sigma \rightrightarrows x\sigma[\pi''_{j_1} \leftarrow d_{j_1}\sigma_{j_1}] \ldots [\pi''_{j_m} \leftarrow d_{j_m}\sigma_{j_m}]$. Thus, in this case, $x\sigma'$ is defined as $x\sigma[\pi''_{j_1} \leftarrow d_{j_1}\sigma_{j_1}] \ldots [\pi''_{j_m} \leftarrow d_{j_m}\sigma_{j_m}]$.

The polymorphic function `choose_seq` below was specified to construct associated subsequences of positions, rules or substitutions. $\texttt{choose\_seq}(\Pi_\pi, \Pi_1, \Gamma_1)$ and $\texttt{choose\_seq}(\Pi_\pi, \Pi_1, \Sigma_1)$ give respectively the subsequences of rules and substitutions associated with Π_π. In particular, `choose_seq` can be used in order to choose the sequence of terms, instantiations of rhs's 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 $\texttt{choose\_seq}(\texttt{sub\_pos}(\Pi_1, \pi\pi'),\ \Pi_1, \{d_1\sigma_1, \ldots, d_n\sigma_n\})$, 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(seq)=0  THEN empty_seq
     ELSIF index(seq1,seq(0)) < seq1'length
           THEN add_first(seq2(index(seq1,seq(0))), choose_seq(rest(seq),seq1,seq2))
           ELSE choose_seq(rest(seq),seq1,seq2)
  ENDIF  MEASURE(length(seq))
```

The function $\texttt{index}(\Pi, \pi)$ above returns the index of the position π in the sequence Π, which is less than the length of Π, if π occurs indeed in Π. Otherwise, it returns the length of Π.

The construction of σ' 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 σ, x and the associated subsequences of substituting terms and positions relative to the necessary update of $x\sigma$. One has, $\texttt{SIGMA}(\sigma, x, \{d_{j_1}\sigma_{j_1}, \ldots, d_{j_m}\sigma_{j_m}\}, \{\pi''_{j_1}, \ldots, \pi''_{j_m}\})$ 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. $\texttt{SIGMAP}(\sigma, \{x_1, \ldots, x_q\}, \{\pi\pi'_1, \ldots, \pi\pi'_q\}, \{d_1\sigma_1, \ldots, d_n\sigma_n\}, \{\pi_1, \ldots, \pi_n\})$, 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 $\texttt{choose\_seq}(\{\pi\pi'_r\pi''_{r,j_1}, \ldots, \pi\pi'_r\pi''_{r,j_{m_r}}\}, \{\pi_1, \ldots, \pi_n\}, \{d_1\sigma_1, \ldots, d_n\sigma_n\})$, which gives the sequence of substituting terms. The latter one is called as $\texttt{complement\_pos}(\pi\pi'_r, \{\pi_1, \ldots, \pi_n\})$, 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 small 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. In fact, this lemma says that, if one compares a position p with a sequence of parallel positions fsp, the complementary positions are obtained from the same positions that are under p.

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

Currently, the whole PVS orthogonal 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.

4 Related work and Conclusions

PVS specifications of non trivial notions and formalizations of results of the theory of term rewriting systems were presented, that are related with the properties of the parallel rewriting reduction and orthogonal rewriting systems. The PVS *theory* for orthogonal TRSs enriches the PVS *theory* trs for TRSs introduced in [6] and available in [14]. The formalization of these properties of orthogonal TRSs are close to the analytical inductive proofs presented in textbooks such as [3] and [4] that in essence are based in the well-known parallel moves lemma which projects parallel reductions over a simple rewriting reduction. These formalizations provide 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 by using standard definitions and proof ideas shown in text books that ease the understanding of them. The last theorem 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 [13] the criterion of weak orthogonality was integrated to ensure confluence applying the certification tool CeTA. Unlike orthogonality, weak orthogonality allows for trivial critical pairs. To the best of our knowledge any complete 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, doi:10.1109/40.768501.

[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 Trans. Design Autom. Electr. Syst.* 11(2), pp. 251–281, doi:10.1145/1142155.1142156.

[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 (2008): *A Formalization of Newman's and Yokouchi Lemmas in a Higher-Order Language*. *Journal of Formalized Reasoning* 1(1), pp. 39–50.

[6] 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, doi:10.1016/j.entcs.2009.07.049.

[7] 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, doi:10.1007/s10817-010-9165-2.

[8] 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, doi:10.1145/322217.322230.

[9] 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.

[10] 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, doi:10.1109/FPL.2005.1515694.

[11] 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, doi:10.1109/FCCM.2008.37.

[12] B. K. Rosen (1973): *Tree-Manipulating Systems and Church-Rosser Theorems*. *J. of the ACM* 20(1), pp. 160–187, doi:10.1145/321738.321750.

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

[14] Theory `trs` (consulted January 2013): *Available in the NASA LaRC PVS library,* `http://shemesh.larc.nasa.gov/fm/ftp/larc/ PVS-library/pvslib.html`.