Formalising Confluence in PVS

Mauricio Ayala-Rincón

Departamentos de Ciência da Computação e Matemática
Universidade de Brasília, Brazil
`ayala@unb.br`

Confluence is a critical property of computational systems which is related with determinism and non ambiguity and thus with other relevant computational attributes of functional specifications and rewriting system as termination and completion. Several criteria have been explored that guarantee confluence and their formalisations provide further interesting information. This work discusses topics and presents personal positions and views related with the formalisation of confluence properties in the Prototype Verification System PVS developed at our research group.

1 Introduction

Syntactic criteria such as avoiding overlapping of rules as well as linearity of rules have been used as a discipline of functional programming which avoids ambiguity. In the context of term rewriting systems (TRSs for short), well-known results such as Newman's Lemma [9], Rosen's Confluence of Orthogonal term rewriting systems [11] as well as the famous Knut-Bendix(-Huet) Critical Pair Lemma [8, 7] are of great theoretical and practical relevance. The first one, guarantees confluence of Noetherian and locally confluent abstract reduction systems; the second one, assures confluence of *orthogonal* term rewriting systems, that are systems that avoid ambiguities generated by overlapping of their rules and whose rules do not allow repetitions of variables in their left-hand side (i.e., left-linear); and, the third one provides local confluence of term rewriting systems whose critical pairs are joinable.

Formalisations in PVS of these confluence criteria provide valuable and precise data about the theory of rewriting (cf. [6], [5], [10]). All mentioned specifications and formalisations are available either at the local site `http://trs.cic.unb.br` or, as part of the NASA PVS libraries, in the theories for abstract reduction systems `ars` and term rewriting systems `trs` that belong to the TRS development, at `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`.

2 Background

It is assumed the reader is familiar with rewriting notations and notions as given in [3] and [4].

2.1 Abstract reduction systems

In the PVS development for TRSs, specifically in the theory `ars`, an abstract reduction system (for short, ARS) is specified as a binary relation R over an uninterpreted type T, `R VAR : PRED[[T,T]]`. This choice facilitates the definition of associated necessary relations through the use of PVS operations for relations such as reversal, subset, union, composition and an operator for iterative applications of compositions. For instance,

- the inverse of the relation is specified as `converse(R);`

- the symmetric closure, `SC(R)`, as `union(R, converse(R)`;
- the reflexive closure, `RC(R)`, as `union(R, =)`;
- the reflexive transitive closure, `RTC(R)`, as `IUnion(LAMBDA n: iterate(R, n))`;
- the equivalence closure, `EC(R)`, as `RTC(SC(R))` etc.

Properties of ARSs are then specified in a very natural manner from these relational basis. For instance, using PVS properties for relations such as well-foundedness, the property of noetherianity, `noetherian?(R)` is specified as the predicate `well_founded?(converse(R))`. Also, the property of confluence, `confluence?(R)`, is specified as

    FORALL x, y, z: RTC(R)(x,y) & RTC(R)(x,z) => joinable?(R)(y,z)

where the predicate `joinable?` is specified as

    joinable?(R)(x,y): bool = EXISTS z: RTC(R)(x,z) & RTC(R)(y, z).

More synthetic specifications might be possible; for instance, the elegant *set-theoretical* definition of confluence, written in the usual rewriting notation as $(^{*}\!\leftarrow \circ \rightarrow^{*}) \subseteq (\rightarrow^{*} \circ {}^{*}\!\leftarrow)$ can be specified straight-forwardly as `subset?(RTC(converse(R)) o RTC(R), RTC(R) o RTC(converse(R)))`, using relation composition, `o`, and the subset predicate, `subset?`.

ARS results were formalised using standard proof techniques as noetherian induction. Among other results on confluence of ARSs, a description of the formalisation of Newman Lemma, specified below, is available in [5]

    Newman: LEMMA noetherian?(R) => (confluent?(R) <=> locally_confluent?(R))

2.2   Term Rewriting Systems

Terms are specified as a data type built from variables over a nonempty uninterpreted type and a signature of function symbols with their respective arities. The arguments of a functional term, headed by a function symbol of the signature, are specified as a finite sequence of terms, of length equal to the arity of the function symbol. Positions of a term `t`, written `posOF(t)`, are finite sequences of naturals specified recursively as in the standard way in the theory of rewriting. Thus, the necessary operations on positions as their concatenation resumes to concatenation of finite sequences of naturals and so, predicates such as disjunct or parallel positions, given by the predicate `parallel?` or for short `||`, are specified as `(NOT p <= q) & (NOT q <= p)`, where `<=` is the sequence prefix relation built as `<=(p, q): bool = (EXISTS (p1: position): q = p o p1)`.

Using these types for terms and positions, it is possible to build the required algebraic properties for terms, positions, subterms and replacement of terms. Namely, the subterm of a term `t` at a valid position `p`, usually written as t_p, is specified recursively navigating through the structure of the term according to the naturals in the position sequence and the arguments of the functional terms inside `t`: `stOF(t: term, (p: positions?(t)))`. Also, the replacement of the subterm at a valid position `p` of a term `s` by another term `t` is built recursively: `replaceTerm(s: term, t: term, (p: positions?(s)))`, that will be abbreviated as `s[t]_p`. These design decisions give rise to algebraic properties that are easily formalised by inductive proofs on these data structures. Among other properties, one has formalisations for:

- Preservation of positions after replacement of subterms either positions of replacement:

    `posOF(s)(p) => posOF(s[t]_p)(p)`

    or parallel positions to the position of replacement:

    `posOF(s)(p) & posOF(s)(q) & p||q => posOF(s[t]_p)(q)`

- Extension of possible new valid positions at the position of replacement after a replacement (where below o stands for the concatenation of sequences):

  `posOF(t)(q) & posOF(s)(p) => posOF(s[t]_p)(p o q)`

- Preservation of replaced terms:

  `posOF(s)(p) & posOF(t)(q) => stOF(s[t]_p,p o q) = stOF(t, q)`

- Associativity of replacement:

  `posOF(s)(p) & posOF(t)(q) => (s[t]_p)[r]_(p o q) = s[t[r]_q]_p`

- Commutativity of replacement at parallel positions:

  `posOF(s)(p) & posOF(s)(q) & p||q => (s[t]_p)[r]_q = (s[r]_q)[t]_p`

Rewriting rules are specified as pairs of terms (l, r) satisfying the usual conditions on rules, that is the left-hand side (lhs) cannot be a variable and the variables occurring in the right-hand side (rhs) should belong to the lhs of the rule:

    rewrite_rule?(l,r): bool = (NOT vars?(l)) & subset?(Vars(r), Vars(l))

After that, it is possible to define the type of rewriting rules as

                    rewrite_rule : TYPE rewrite_rule?

and then, a TRS is given as a set of rewriting rules `set[rewrite_rule]`.

Substitutions are built as objects of type `[V -> term]`, where `V` is a countably infinite set of variables and such that their domain is finite, that is `Sub?(sig):  bool = is_finite(Dom(sig))`, where `Dom(sig):  set[(V)] = {x:  (V) | sig(x) /= x}` . The type of substitutions is given as `Sub: TYPE = (Sub?)`. From this point, renaming, variants, composition of substitutions and homomorphic extensions of substitutions, `ext(sigma)`, are easily built as well as a series of necessary substitution properties formalised.

With these elements of formal design it is possible to define the reduction relation from a set of rewriting rules say `E`:

```
reduction?(E)(s,t): bool =
    EXISTS ( (e | member(e, E)), sigma, (p: positions?(s))) :
                  stOF(s, p) = ext(sigma)(lhs(e)) &
                                t = s[ext(sigma)(rhs(e))]_p
```

Immediately, it is possible to prove that this relation is *closed under substitutions* and *compatible with contexts*.

After that, a predicate for critical pairs of a TRS E is built, `CP?(E)`, and then the most famous result on confluence of TRSs, that is the Critical Pair Lemma is formalised as described in [6].

```
CP: THEOREM FORALL E: locally_confluent?(reduction?(E)) <=>
      (FORALL s, t) : CP?(E)(s, t) => joinable?(reduction?(E))(s,t)
```

Since the reduction relation built from a set of rewriting rules inherits by parameterisation, all properties of ARSs in the `ars` development, it is possible to apply Newman's Lemma in order to formally infer confluence of a terminating TRS all whose critical pairs are joinable.

The design decisions taken in the specification of ARSs and TRSs were satisfactory to accomplish one of our main objectives in this formalisation, that is indeed maintaining formal proofs as close as

possible to the analytical proofs presented in textbooks. In fact, diagrammatic didactical artefacts (used in papers and textbook) representing rewriting properties used in the proofs as commutation diagrams for confluence, local-confluence etc, and those ones used for representing peaks valleys and overlap situations in the analysis of the Critical Pair criterion can be also conducted when reasoning about our formalisations.

In particular, the formalisation of the Critical Pair Lemma follows the textbook proof organisation which is based on the analysis of the possible peaks when trying to obtain local confluence. These peaks can be originated from simultaneous reductions at parallel positions, which are trivially joinable, or from reductions at nested positions, which give rise to either *critical* or *non-critical overlaps*.

A peak, from a critical overlap can be easily verified to join, by proving that it corresponds to an instance of a critical pair and using the assumption that critical pairs are joinable.

The non-critical overlaps are the interesting ones. A such peak is originated by application of rules $l \to r$ and $g \to d$ with substitution σ, assuming these rules have not common variables which is possible by renaming of rules. Supposing $l\sigma$ occurs in the dominating position of the overlap, one can focus on the analysis of the *joinability* of the peak $r\sigma \leftarrow l\sigma \to l\sigma[d\sigma]_{p\circ q}$, where p is a variable position in l, say $l_p = x$, and q the position in $x\sigma$ in which $g\sigma$ occurs.

Thus, all that is solved by the careful construction of a new substitution, σ' such that it modifies σ mapping $x \mapsto x\sigma[d\sigma]_q$ and maintains the images of all other variables in the domain of σ as those mapped by σ.

In the sequel, by a like *"uniform reduction sequence"* one has that $r\sigma \to^* r\sigma'$ and $l\sigma[d\sigma]_{p\circ q} \to^* l\sigma'$; the former is done as $r\sigma \to r\sigma[d\sigma]_{q_1 \circ q} \to \cdots \to r\sigma[d\sigma]_{q_1 \circ q} \cdots [d\sigma]_{q_n \circ q} = r\sigma'$, where $\{q_1, \ldots, q_n\}$ is the set of positions of r in which x occurs, and the latter is done as $l\sigma[d\sigma]_{p\circ q} \to l\sigma[d\sigma]_{p\circ q}[d\sigma]_{p_1 \circ q} \to \cdots \to l\sigma[d\sigma]_{p\circ q}[d\sigma]_{p_1 \circ q} \cdots [d\sigma]_{p_m \circ q} = l\sigma'$, where $\{p\} \cup \{p_1, \ldots, p_n\}$ is the set of positions of l in which x occurs.

So joinability is concluded by the application of rule $l \to r$ with substitution σ'. See for instance the proof in Chapters 6 or 2 of respectively [3] or [4].

The formalisation, under the design choices previously mentioned, requires the construction of elements that guarantee specialised properties, such as the instantiation of a critical pair, built from the rewriting rules, that corresponds to a critical overlap as well as the substitution σ' for a non-critical overlap. For the latter, it is necessary an inductive proof (using auxiliary lemmas) on the cardinality of the sets of positions $\{q_1, \ldots, q_n\}$ and $\{p_1, \ldots, p_m\}$ for concluding that $l\sigma$ and $r\sigma$ rewrite into $l\sigma'$ and $r\sigma'$, respectively.

3   Formalising the algebra of parallel rewriting and orthogonality

Rosen's confluence of orthogonal TRSs [11] is a challenging formalisation. The classical proof is based on the Parallel Moves lemma: essentially, what is necessary is to prove that under the hypothesis of orthogonality, the associated parallel reduction relation holds the diamond property.

Intuitively, the analytical proof requires only the comprehension of properties for the notion of the parallel reduction relation, but the intuition of parallel rewriting is usually explained through the like "uniform reduction" as in the analysis of the Critical Pair criterion, which in fact refers to sequential rewriting. So, any formalisation following the classical approach would require an explicit construction of such that parallel relation as well as the specialised description and formalisation of its specific algebraic properties.

The notion of parallel reduction depends on sets of triplets of valid positions, rules and substitutions,

positions in which sequential replacements are simultaneously applied according to the instantiation of the rules with the associated substitutions. Because of this dependence, two design approaches are possible: either using sets of triplets of positions, rules and substitutions or sets of (finite) and coordinated sequences of positions Π, rules Γ and substitutions Σ. We opted by the last design alternative since we believe it is closer to implementations in programming languages and also because PVS offers libraries with translations (and their necessary formalised properties) between data structures such as sets, lists and finite sequences.

The parallel rewriting reduction relation built from a set of rewriting rules E, that in classical notation is written as $s \rightrightarrows t$, is specified as the relation `parallel_reduction?(E)(s,t)` below, using a parallel replacement operator, `replace_par_pos`, that is recursively specified from the sequential `replaceTerm` operator, and through an auxiliary relation `parallel_reduction_fix?(E)`.

```
parallel_reduction_fix?(E)(s,t, (fsp: SPP(s))): bool =
   EXISTS ((fse | member(fse, E)), fss) :
           fsp'length = fse'length AND fsp'length = fss'length
           AND subtermsOF(s,fsp) = sigma_lhs(fss, fse)
           AND t = replace_par_pos(s, fsp, sigma_rhs(fss, fse))


parallel_reduction?(E)(s,t): bool =
   EXISTS (fsp: SPP(s)): parallel_reduction_fix?(E)(s,t,fsp)
```

`fsp`, `fse` and `fss` correspond to the sequences $\Pi = [p_1, \ldots, p_n]$, $\Gamma = [(l_1, r_1), \ldots, (l_n, r_n)]$ and $\Sigma = [\sigma_1, \ldots, \sigma_n]$ of positions, rewriting rules and substitutions used in the parallel rewriting. `fsp` is a sequence of parallel positions of s obtained by its type dependency, that is `SSP(s)`. `fse` is a sequence of equations in the rewriting system given by `member(fse, E)`, and `fss` is a sequence of substitutions. The required coordination of triplets of associated positions, rules and substitutions is directly obtained by using the corresponding indexation in the respective sequences, that is the same index. The condition `subtermsOF(s,fsp) = sigma_lhs(fss, fse)` equals the condition that for all valid index i of these sequences, $s_{p_i} = l_i \sigma_i$ and the condition `t= replace_par_pos(s, fsp, sigma_rhs(fss, fse))` equals t to the desired parallel contractum, that is s replacing the $l_i \sigma_i$'s by the $r_i \sigma_i$'s.

In a parallel peak, say $t \Leftleftarrows s \rightrightarrows u$, using positions, equations and substitutions say $(\Pi_k, \Gamma_k, \Sigma_k)$ for $k = 1, 2$, as in the case of the Critical Pair Lemma, the interesting cases are those of non-critical overlaps. Without loss of generality, at some position $q \in \Pi_1$ one has all positions $p_1, ..., p_n$ in Π_2 below q. On the one side, $s_q = l\sigma$ and $t_q = r\sigma$, where (l, r) is the rewriting rule in Γ_1 and σ the substitution in Γ_1 associated with the position q in Π_1. On the other side, one has $s_q \rightrightarrows u_q$ by parallel reduction at positions $p_1, \ldots p_n$ below q accordingly to the same equations and substitutions in the triplet $(\Pi_2, \Gamma_2, \Sigma_2)$. Let (Π', Γ', Σ') denote the triplet of this last parallel reduction step, then the parallel peak $r\sigma \Leftleftarrows l\sigma \rightrightarrows u_q$ is an instance of the Parallel Moves Lemma. See details in Chapter 4.3 of [4] or 6.4 of [3], for instance.

Since in such a parallel peak $r\sigma \Leftleftarrows l\sigma \rightrightarrows t$ all overlaps are non critical, one should prove that parallel reducing each σ-instance of a variable in l, say x at position p one has $l\sigma_p = x\sigma \rightrightarrows x\sigma' = t_p$, where the substitution σ' is built by reducing in parallel all occurrences of σ-instantiated variables in $l\sigma$ uniformly, which is possible by left-linearity assumption. Thus, $r\sigma \rightrightarrows r\sigma'$ and $t \rightrightarrows l\sigma'$, that allows concluding the joinability of the peak.

Despite in the theory the adaptation of sequential properties for parallel replacement might be intuitively clear, in the PVS development the necessary specialised algebraic properties should be formalised. Let `s[T]`$\_\Pi$ denote the parallel replacement of terms in the sequence of terms T at valid parallel positions Π. A few of those properties are included below.

- Preservation of positions after replacement of subterms either positions of replacement:

  `posOF(s)(`Π`) => posOF(s[T]_`Π`)(`Π`)`

  or parallel positions to the position of replacement:

  `posOF(s)(`Π`) & posOF(s)(`Π'`) & `Π'`||`Π` => posOF(s[t]_p)(`Π'`)`

- Invariance under composition of parallel replacement at parallel sequences of positions:

  `posOF(s)(`Π_1`) & posOF(s)(`Π_2`) & `Π_1`||`Π_2` =>`

  $$(s[T\_1]\_\Pi_1)[T\_2]\_\Pi_2 = (s[T\_1 \text{ o } T\_2]\_\Pi_1 \circ \Pi_2$$

- Commutativity of parallel replacement at parallel sequences of positions:

  `posOF(s)(`Π_1`) & posOF(s)(`Π_2`) & `Π_1`||`Π_2` =>`

  $$(s[T\_1]\_\Pi_1)[T\_2]\_\Pi_2 = (s[T\_2]\_\Pi_2)[T\_1]\_\Pi_1$$

The formalisation of confluence of orthogonal TRS proceeds by inductive proof techniques taking care of the specificities of the algebra of parallel positions, replacement and rewriting.

4   Conclusions and future work

The development of these formalisations on confluence of TRSs brought out several lessons.

From the theoretical point of view, the main observation is that despite the intuitive notion of "uniform reduction", that is used to provide intuition about the proof of the Parallel Moves Lemma, induces to believe that the extension is obvious, a specialised development of the theory of parallel reduction and its algebraic properties is necessary. And extending sequential to parallel rewriting formalisations is not trivial. A preliminary thoughtful analysis would be always necessary in order to estimate accurately the real complexity and the necessary effort of formalisations in any context, mostly when the proposed development appears to be a simple extension of those yet available. The second lesson has to do with the investment of enough time for fine tuning design decisions since they influence the effort required in proofs.

The main lesson is that through this kind of exercise, our comprehension of the theory becomes more refined, providing a better support for the formal analysis of further related developments, as well as a more realistic insight about the possible adaptation or reuse of previous specifications and proofs. Developments in progress include use of such techniques in other contexts as the nominal syntax approach of rewriting (cf. [1] [2]).

References

[1] M. Ayala-Rincón, M. Fernández, M. J. Gabbay & A. C. Rocha Oliveira (2015): *Checking Overlaps of Nominal Rewriting Rules*. In: *Pre-proc. LSFA*, pp. 199–2014. Available at `https://www.mat.ufrn.br/~LSFA2015/preproceedings.pdf`.

[2] M. Ayala-Rincón, M. Fernández & A. C. Rocha Oliveira (2015): *Completeness in PVS of a Nominal Unification Algorithm*. In: *Pre-proc. LSFA*, pp. 19–34. Available at `https://www.mat.ufrn.br/~LSFA2015/preproceedings.pdf`.

[3] F. Baader & T. Nipkow (1998): *Term Rewriting and* All That. Cambridge University Press, doi:10.1017/CBO9781139172752.

[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, doi:10.6092/issn.1972-5787/1347.

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

[7] G. Huet (1981): *A complete proof of correctness of the Knuth-Bendix completion algorithm*. Journal of Computer and Systems Sciences 23(1), pp. 11–21, doi:10.1016/0022-0000(81)90002-7.

[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., doi:10.1016/B978-0-08-012975-4.50028-X.

[9] M. H. A. Newman (1942): *On theories with a combinatorial definition of "equivalence"*. Ann. of Math. 43(2), pp. 223–243, doi:10.2307/1968867.

[10] A. C. Rocha Oliveira & M. Ayala-Rincón (2013): *Formalizing the Confluence of Orthogonal Rewriting Systems*. *CoRR* abs/1303.7335, doi:10.4204/EPTCS.113.14. Available at http://arxiv.org/abs/1303.7335.

[11] 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.