Formalizing Rewriting and Termination in PVS

Mauricio Ayala-Rincón

Universidade de Brasília (UnB)

Brasília D.F., Brazil

Research funded by

Brazilian Research Agencies: CNPq, CAPES and FAPDF

Joint short course with César Muñoz

International School on Rewriting ISR 2018
Universidad Javeriana Cali, Colombia - Aug 1^{st} 2018

Universidade de Brasília

Talk's Plan

The Prototype Verification System - PVS

PVS is a verification system, developed by the SRI International Computer Science Laboratory, which consists of

1. a *specification language*:
   - based on *higher-order logic*;
   - a type system based on Church's simple theory of types augmented with *subtypes* and *dependent types*.
2. an *interactive theorem prover*:
   - based on **sequent calculus**; that is, goals in PVS are sequents of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite sequences of formulae, with the usual Gentzen semantics.

The Prototype Verification System - PVS — Libraries

- NASA LaRC PVS library includes
  - Structures, analysis, algebra, Graphs, Digraphs,
  - real arithmetic, floating point arithmetic, groups, interval arithmetic,
  - linear algebra, measure integration, metric spaces,
  - orders, probability, series, sets, topology,
  - term rewriting systems, unification, etc. etc.

The Prototype Verification System - PVS — Sequent calculus

- Sequents of the form: $\Gamma \vdash \Delta$.
  - Interpretation: from Γ one obtains Δ.
  - $A_1, A_2, \ldots, A_n \vdash B_1, B_2, \ldots, B_m$ interpreted as $A_1 \wedge A_2 \wedge \ldots \wedge A_n \vdash B_1 \vee B_2 \vee \ldots \vee B_m$.
- Inference rules
  - Premises and conclusions are simultaneously constructed:

$$\frac{\Gamma \vdash \Delta}{\Gamma' \vdash \Delta'}$$

- Goal: $\vdash \Delta$.

Sequent calculus in PVS

- Representation of $A_1, A_2, ..., A_n \vdash B_1, B_2, ..., B_m$:

```
[-1]  A1
  :
[-n]  An
|----------
 [1]  B1
  :
 [n]  Bn
```

- Proof tree: each node is labelled by a sequent.
- A PVS proof command corresponds to the application of an inference rule.
  - In general:

$$\frac{\Gamma \vdash \Delta}{\Gamma_1 \vdash \Delta_1 ... \Gamma_n \vdash \Delta_n} \textbf{ (Rule Name)}$$

Some inference rules in PVS

- Structural:

$$\boxed{\frac{\Gamma_2 \vdash \Delta_2}{\Gamma_1 \vdash \Delta_1}\ \mathbf{(W)}, \text{ if } \Gamma_1 \subseteq \Gamma_2 \text{ and } \Delta_1 \subseteq \Delta_2}$$

- Propositional:

$$\boxed{\frac{\Gamma, A \vdash A, \Delta}{}\ \mathbf{(Ax)}} \qquad \boxed{\frac{\Gamma, \mathit{FALSE} \vdash \Delta}{}\ \mathbf{(FALSE \vdash)}}$$

$$\boxed{\frac{\Gamma \vdash \mathit{TRUE}, \Delta}{}\ \mathbf{(\vdash TRUE)}}$$

Some inference rules in PVS

- Cut:
  - Corresponds to the `case` and `lemma` proof commands.

$$\boxed{\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta \qquad \Gamma \vdash A, \Delta}\ \textbf{(Cut)}}$$

- Conditional: `IF-THEN-ELSE`.

$$\boxed{\frac{\Gamma, \mathbf{IF}(A, B, C) \vdash \Delta}{\Gamma, A, B \vdash \Delta \qquad \Gamma, C \vdash A, \Delta}\ \textbf{(IF} \vdash \textbf{)}}$$

$$\boxed{\frac{\Gamma \vdash \mathbf{IF}(A, B, C)\Delta}{\Gamma, A \vdash B, \Delta \qquad \Gamma \vdash A, C, \Delta}\ \textbf{(} \vdash \textbf{IF)}}$$

Gentzen Calculus

sequents:

$$\underset{\text{antecedent}}{\underset{\uparrow}{\Gamma}} \quad \Rightarrow \quad \underset{\text{succedent}}{\underset{\uparrow}{\Delta}}$$

Gentzen Calculus

Table: Rules of deduction *à la* Gentzen for predicate logic

left rules	right rules
Axioms:	
$\Gamma, \varphi \Rightarrow \varphi, \Delta \quad (Ax)$	$\bot, \Gamma \Rightarrow \Delta \quad (L_\bot)$
Structural rules:	
$\dfrac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \; (LWeakening)$	$\dfrac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi} \; (RWeakening)$
$\dfrac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \; (LContraction)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi} \; (RContraction)$

Gentzen Calculus

Table: Rules of deduction *à la* Gentzen for predicate logic

left rules	right rules
Logical rules:	
$\dfrac{\varphi_{i\in\{1,2\}}, \Gamma \Rightarrow \Delta}{\varphi_1 \wedge \varphi_2, \Gamma \Rightarrow \Delta}\ (L_\wedge)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}\ (R_\wedge)$
$\dfrac{\varphi, \Gamma \Rightarrow \Delta \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta}\ (L_\vee)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi_{i\in\{1,2\}}}{\Gamma \Rightarrow \Delta, \varphi_1 \vee \varphi_2}\ (R_\vee)$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta}\ (L_\rightarrow)$	$\dfrac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi}\ (R_\rightarrow)$
$\dfrac{\varphi[x/t], \Gamma \Rightarrow \Delta}{\forall_x \varphi, \Gamma \Rightarrow \Delta}\ (L_\forall)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi[x/y]}{\Gamma \Rightarrow \Delta, \forall_x \varphi}\ (R_\forall), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$
$\dfrac{\varphi[x/y], \Gamma \Rightarrow \Delta}{\exists_x \varphi, \Gamma \Rightarrow \Delta}\ (L_\exists), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$	$\dfrac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{\Gamma \Rightarrow \Delta, \exists_x \varphi}\ (R_\exists)$

Gentzen Calculus

Derivation of the Peirce's law:

$$\dfrac{\dfrac{(R_\rightarrow)\,\dfrac{(RW)\,\dfrac{\varphi \Rightarrow \varphi \;\; (Ax)}{\varphi \Rightarrow \varphi, \psi}}{\Rightarrow \varphi, \varphi \rightarrow \psi} \qquad \varphi \Rightarrow \varphi \;\; (Ax)}{(\varphi \rightarrow \psi) \rightarrow \varphi \Rightarrow \varphi}\,(R_\rightarrow)}{\Rightarrow ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi}\,(L_\rightarrow)$$

Gentzen Calculus

Cut rule:

$$\boxed{\frac{\Gamma \Rightarrow \Delta, \varphi \quad \varphi, \Gamma' \Rightarrow \Delta'}{\Gamma\Gamma' \Rightarrow \Delta\Delta'} \; (Cut)}$$

Gentzen Calculus - dealing with negation: c-equivalence

$$\varphi, \Gamma \Rightarrow \Delta \text{ one-step c-equivalent } \Gamma \Rightarrow \Delta, \neg\varphi$$

$$\Gamma \Rightarrow \Delta, \varphi \text{ one-step c-equivalent } \neg\varphi, \Gamma \Rightarrow \Delta$$

The c-equivalence is the equivalence closure of this relation.

Lemma (One-step c-equivalence)

(i) $\vdash_G \varphi, \Gamma \Rightarrow \Delta$, *iff* $\vdash_G \Gamma \Rightarrow \Delta, \neg\varphi$;

(ii) $\vdash_G \neg\varphi, \Gamma \Rightarrow \Delta$, *iff* $\vdash_G \Gamma \Rightarrow \Delta, \varphi$.

Gentzen Calculus - dealing with negation

Proof.

(i) **Necessity**:

$$\dfrac{\dfrac{\varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta, \bot}\ (\mathrm{RW})}{\Gamma \Rightarrow \Delta, \neg\varphi}\ (\mathrm{R}_{\rightarrow})$$

Sufficiency:

$$\dfrac{(\mathrm{LW})\ \dfrac{\Gamma \Rightarrow \Delta, \neg\varphi}{\varphi, \Gamma \Rightarrow \Delta, \neg\varphi} \qquad \dfrac{(Ax)\ \varphi, \Gamma \Rightarrow \Delta, \varphi \qquad \bot, \varphi, \Gamma \Rightarrow \Delta\ (L_{\bot})}{\neg\varphi, \varphi, \Gamma \Rightarrow \Delta}\ (\mathrm{L}_{\rightarrow})}{\varphi, \Gamma \Rightarrow \Delta}\ (\mathrm{Cut})$$

Gentzen Calculus - dealing with negation

(ii) **Necessity**:

$$
\dfrac{
\dfrac{
(R_\to)\ \dfrac{
(L_\to)\ \dfrac{
(R_\to)\ \dfrac{(Ax)\ \varphi, \Gamma \Rightarrow \Delta, \varphi, \varphi, \bot}{\Gamma \Rightarrow \Delta, \varphi, \varphi, \neg\varphi}
\qquad \bot, \Gamma \Rightarrow \Delta, \varphi, \varphi\ (L_\bot)
}{\neg\neg\varphi, \Gamma \Rightarrow \Delta, \varphi, \varphi}
}{\Gamma \Rightarrow \Delta, \varphi, \neg\neg\varphi \to \varphi}
}{}
\qquad
\dfrac{
\dfrac{
\dfrac{\neg\varphi, \Gamma \Rightarrow \Delta}{\neg\varphi, \Gamma \Rightarrow \Delta, \varphi, \bot}\ (RW)
}{\Gamma \Rightarrow \Delta, \varphi, \neg\neg\varphi}\ (R_\to)
\qquad \varphi, \Gamma \Rightarrow \Delta, \varphi\ (Ax)
}{\neg\neg\varphi \to \varphi, \Gamma \Rightarrow \Delta, \varphi}\ (L_\to)
}{\Gamma \Rightarrow \Delta, \varphi}\ (\textsc{Cut})
$$

Sufficiency:

$$
\dfrac{\Gamma \Rightarrow \Delta, \varphi \qquad \bot, \Gamma \Rightarrow \Delta}{\neg\varphi, \Gamma \Rightarrow \Delta}\ (L_\to)
$$

□

Universidade de Brasília

Summary - Gentzen Deductive Rules vs Proof Commads

Table: Structural Left Rules vs Proof Commands

Structural left rules	PVS commands
$\dfrac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}\ (LWeakening)$	$\dfrac{\varphi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ (hide)$
$\dfrac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}\ (LContraction)$	$\dfrac{\varphi, \Gamma \vdash \Delta}{\varphi, \varphi, \Gamma \vdash \Delta}\ (Copy)$

Summary - Gentzen Deductive Rules vs Proof Commads

Table: Structural Right Rules vs Proof Commands

Structural right rules	PVS commands
$\dfrac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$ $(RWeakening)$	$\dfrac{\Gamma \vdash \Delta, \varphi}{\Gamma \vdash \Delta}$ $(Hide)$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}$ $(RContraction)$	$\dfrac{\Gamma \vdash \Delta, \varphi}{\Gamma \vdash \Delta, \varphi, \varphi}$ $(Copy)$

Summary - Gentzen Deductive Rules vs Proof Commads

Table: Logical Left Rules vs Proof Commands

left rules	PVS commands
$\dfrac{\varphi_1, \varphi_2, \Gamma \Rightarrow \Delta}{\varphi_1 \wedge \varphi_2, \Gamma \Rightarrow \Delta}\ (L_\wedge)$	$\dfrac{\varphi_1 \wedge \varphi_2, \Gamma \vdash \Delta}{\varphi_{i\in\{1,2\}}, \Gamma \vdash \Delta}\ (\mathit{Flatten})$
$\dfrac{\varphi, \Gamma \Rightarrow \Delta \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta}\ (L_\vee)$	$\dfrac{\varphi \vee \psi, \Gamma \vdash \Delta}{\varphi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}\ (\mathit{Split})$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \psi, \Gamma \Rightarrow \Delta}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta}\ (L_\rightarrow)$	$\dfrac{\varphi \rightarrow \psi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi \quad \psi, \Gamma \vdash \Delta}\ (\mathit{Split})$
$\dfrac{\varphi[x/t], \Gamma \Rightarrow \Delta}{\forall_x \varphi, \Gamma \Rightarrow \Delta}\ (L_\forall)$	$\dfrac{\forall_x \varphi, \Gamma \vdash \Delta}{\varphi[x/t], \Gamma \vdash \Delta}\ (\mathit{Instantiate})$
$\dfrac{\varphi[x/y], \Gamma \Rightarrow \Delta}{\exists_x \varphi, \Gamma \Rightarrow \Delta}\ (L_\exists), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$	$\dfrac{\exists_x \varphi, \Gamma \vdash \Delta}{\varphi[x/y], \Gamma \vdash \Delta}\ (\mathit{Skolem}), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$

Summary - Gentzen Deductive Rules vs Proof Commads

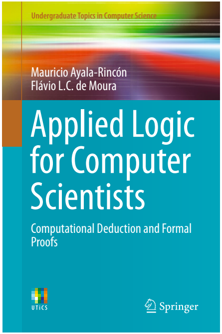
Table: Logical Right Rules vs Proof Commands

right rules	PVS commands
$\dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}\ (R_\wedge)$	$\dfrac{\Gamma \vdash \Delta, \varphi \wedge \psi}{\Gamma \vdash \Delta, \varphi \quad \Gamma \vdash \Delta, \psi}\ (\mathit{Split})$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi_{i \in \{1,2\}}}{\Gamma \Rightarrow \Delta, \varphi_1 \vee \varphi_2}\ (R_\vee)$	$\dfrac{\Gamma \vdash \Delta, \varphi_1 \vee \varphi_2}{\Gamma \vdash \Delta, \varphi_1, \varphi_2}\ (\mathit{Flatten})$
$\dfrac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi}\ (R_\rightarrow)$	$\dfrac{\Gamma \vdash \Delta, \varphi \rightarrow \psi}{\varphi, \Gamma \vdash \Delta, \psi}\ (\mathit{Flatten})$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi[x/y]}{\Gamma \Rightarrow \Delta, \forall_x \varphi}\ (R_\forall), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$	$\dfrac{\Gamma \vdash \Delta, \forall_x \varphi}{\Gamma \vdash \Delta, \varphi[x/y]}\ (\mathit{Skolem}), \quad y \notin \mathtt{fv}(\Gamma, \Delta)$
$\dfrac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{\Gamma \Rightarrow \Delta, \exists_x \varphi}\ (R_\exists)$	$\dfrac{\Gamma \vdash \Delta, \exists_x \varphi}{\Gamma \vdash \Delta, \varphi[x/t]}\ (\mathit{Instantiate})$

Summary - Completing the GC vs PVS rules

	(hide)	(copy)	(flatten)	(split)	(Skolem)	(Inst)	(lemma) (case)
(LW)	×						
(LC)		×					
($L_\wedge$)			×				
($L_\vee$)				×			
($L_\rightarrow$)				×			
($L_\forall$)						×	
($L_\exists$)					×		
(RW)	×						
(RC)		×					
($R_\wedge$)				×			
($R_\vee$)			×				
($R_\rightarrow$)			×				
($R_\forall$)					×		
($R_\exists$)						×	
(Cut)							×

References



2017

Logic for CS with applications to algorithm verification and details on the relations between Gentzen DN and SC rules and PVS proof commands

Formalizing Rewriting Properties

Dealing with HO variables, quantifying binary relations, and induction:

Theorem (CR vs C)

Confluence and CR are equivalent properties

Abstract Reduction Systems - Binary relations

```
relations_closure[T : TYPE] : THEORY
BEGIN
   IMPORTING    orders@closure_ops[T],    sets_lemmas[T]
                         ⋮
   S, R: VAR pred[[T, T]]
   n: VAR nat
   p: VAR posnat
                         ⋮
   RC(R): reflexive = union(R, =)
   SC(R): symmetric = union(R, converse(R))
   TC(R): transitive = IUnion(LAMBDA p: iterate(R, p))
   RTC(R): reflexive_transitive = IUnion(LAMBDA n: iterate(R, n))
   EC(R): equivalence = RTC(SC(R))
                         ⋮
END relations_closure
```

Abstract Reduction Systems

change_to_TC : LEMMA transitive_closure(R) = TC(R)

R_subset_TC :LEMMA subset?(R, TC(R))

TC_converse: LEMMA TC(converse(R)) = converse(TC(R))

TC_idempotent : LEMMA TC(TC(R)) = TC(R)

TC_characterization : LEMMA transitive?(S) $\Leftrightarrow$ (S = TC(S))

Abstract Reduction Systems - PVS Theory

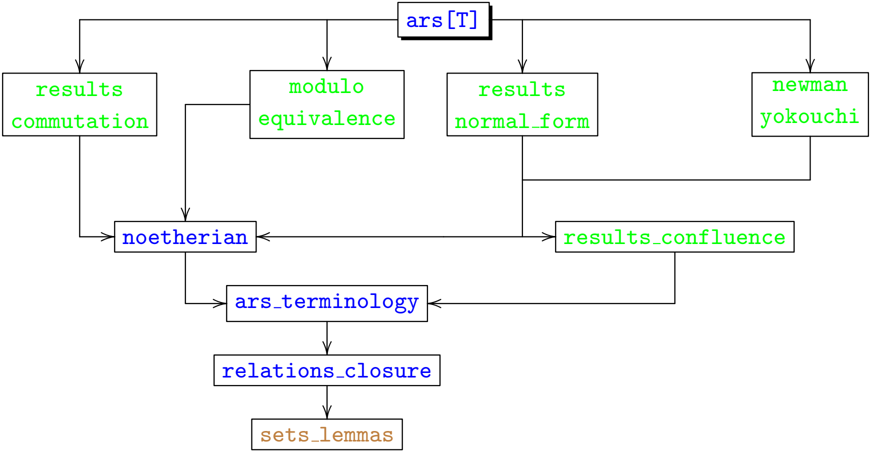


Figure: Hierarchy of the ars theory (Av. at NASA LaRC PVS library)

Case of study - Newman's Lemma

noetherian?(R): bool = well_founded?(converse(R))

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

locally_confluent?(R): bool =
FORALL x, y, z: R(x,y) & R(x,z) ⇒ joinable?(R)(y,z)

confluent?(R): bool =
FORALL x, y, z: RTC(R)(x,y) & RTC(R)(x,z) ⇒ joinable?(R)(y,z)

Newman_lemma: THEOREM
noetherian?(R) ⇒ (confluent?(R) ⇔ locally_confluent?(R))

Case of study - Newman's Lemma

- Hands in the dough -

PVS files with Newman's Lemma formalization downloadable as `NewmanLemma.tgz`

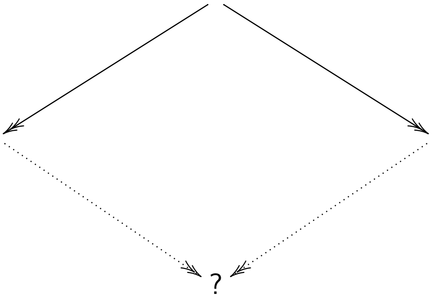


Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

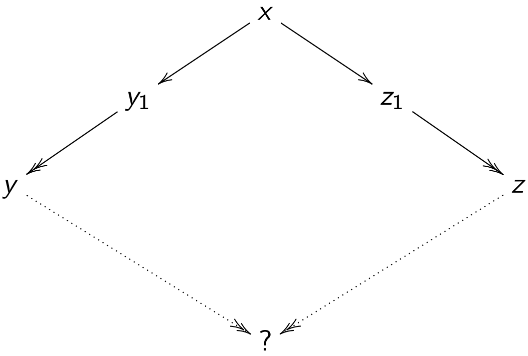


Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

$$
\begin{tikzcd}
 & & x \arrow[dl] \arrow[dr] & & \\
 & y_1 \arrow[dl] \arrow[dr] & L.C. & z_1 \arrow[dl] \arrow[dr] & \\
y \arrow[drr, dotted] & & \exists u & & z \arrow[dll, dotted] \\
 & & ? & &
\end{tikzcd}
$$

Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

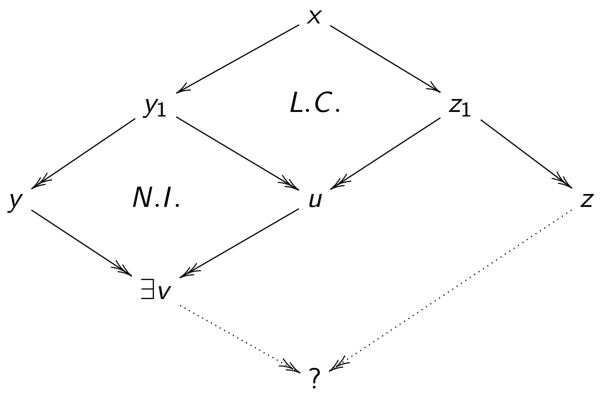


Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

$$
\begin{tikzcd}
 & & x \arrow[dl] \arrow[dr] & & \\
 & y_1 \arrow[dl] \arrow[dr] & L.C. & z_1 \arrow[dl] \arrow[dr] & \\
y \arrow[dr] & & u \arrow[dl] & & z \arrow[ddll] \\
 & v \arrow[dr] & N.I. & & \\
 & & \exists w & &
\end{tikzcd}
$$

Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

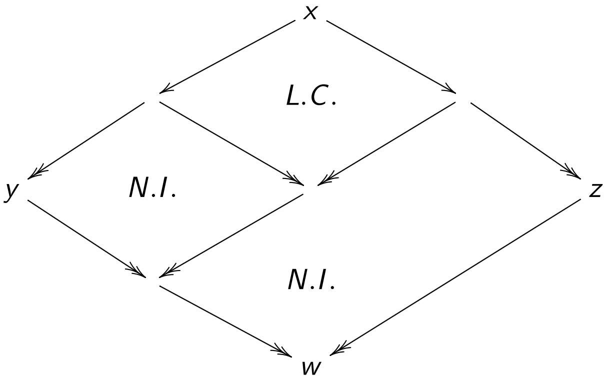


Figure: Proof's Sketch of Newman's Lemma

Case of study - Newman's Lemma

A few used lemmas:

```
R_subset_RC : LEMMA subset?(R, RC(R))
iterate_RTC: LEMMA FORALL n : subset?(iterate(R, n), RTC(R))
R_is_Noet_iff_TC_is: LEMMA noetherian?(R) ⇔ noetherian?(TC(R))
R_subset_TC :LEMMA subset?(R, TC(R))
```

```
noetherian_induction: LEMMA
  (FORALL (R: noetherian, P):
   (FORALL x:
    (FORALL y: TC(R)(x, y) ⇒ P(y))
     ⇒ P(x))
   ⇒
    (FORALL x: P(x)))
```

trs *Theory - Hierarchy*

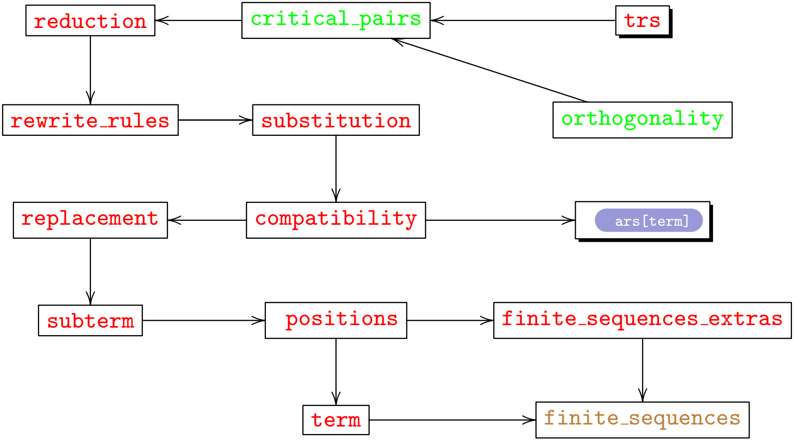


Figure: Hierarchy of the trs *theory*

TRS specification - Terms

The set of terms

```
term[variable: TYPE+, symbol: TYPE+] : DATATYPE
BEGIN

IMPORTING arity[symbol]

vars(v: variable): vars?

app(f:symbol,
    args:{args:finite_sequence[term] | length(args)=arity(f)}): app?

END term
```

TRS specification - Other key basic concepts

Positions and Subterms

- The *set of positions* of the term t, denoted by $Pos(t)$, is inductively defined as follows:
  - *(a)* If $t = x \in V$, then $Pos(t) := \epsilon$, where ϵ denotes the empty string.
  - *(b)* If $t = f(t_1, \cdots, t_n)$, then

$$Pos(t) := \{\epsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in Pos(t_i)\}$$

- The *subterm* of a term s at position $p \in Pos(s)$, denoted by $s|_p$, is inductively defined on the length of p as follows:

$$\begin{array}{rcl} s|_\epsilon & := & s \\ f(s_1, \ldots, s_n)|_{iq} & := & s_i|_q \end{array}$$

TRS specification - Replacement

Replacing the subterm of s at position $p \in Pos(s)$ by t: $s[p \leftarrow t]$

```
replaceTerm(t: term, s: 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(t, st(i-1), q), st,i-1) IN
           app(f(s), rst)
      ENDIF)
   MEASURE length(p)
```

Usefull properties

Let s, t, r be terms. If p and q are parallel positions in s, then

(a) $s[p \leftarrow t]|_q = s|_q$ persistence

(b) $s[p \leftarrow t][q \leftarrow r] = s[q \leftarrow r][p \leftarrow t]$ commutativity

TRS specification - Substitution and Renaming

Substitution

(a) The substitutions are built as functions from variables to terms

```
sig:  [V -> term]
```

whose domain is finite:

```
Sub?(sig):  bool = is_finite(Dom(sig))
```

(b) The homomorphic extension `ext(sig)` of a substitution `sig` is specified inductively over the structure of terms.

Renaming

```
Ren?(sig): bool = subset?(Ran(sig),V) &
                  (bijective?[(Dom(sig)),(Ran(sig))])(sig)
```

TRS specification - Rewrite Rules and Reduction Relation

Rewrite Rules

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

rewrite_rule: TYPE = (rewrite_rule?)
```

Reduction Relation

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

Lemma

Let `E` be a set of rewrite rules. The reduction relation `reduction?(E)` is closed under substitutions and compatible with operations (structure of terms).

TRS specification - Critical Pairs

Critical Pairs - Analytic Definition

Let $l_i \to r_i$, $i = 1, 2$, be two rules whose "variables have been renamed" such that $Var(l_1) \cap Var(l_2) = \emptyset$. Let $p \in Pos(l_1)$ be such that $l_1|_p$ is not a variable and let $\sigma = mgu(l_1|_p, l_2)$. This determines a *critical pair* $\langle t_1, t_2 \rangle$:

$$\begin{array}{lcl} t_1 & = & \sigma(r_1) \\ t_2 & = & \sigma(l_1)[p \leftarrow \sigma(r_2)] \end{array}$$

Critical Pairs - Specification

```
CP?(E)(t1, t2): bool =
 EXISTS (sigma, rho, (e1 | member(e1, E)), (e2p | member(e2p, E)),
       (p: positions?(lhs(e1)))):
        LET e2 = (# lhs := ext(rho)(lhs(e2p)),
                    rhs := ext(rho)(rhs(e2p)) #) IN
        disjoint?(Vars(lhs(e1)),Vars(lhs(e2)))                        &
        NOT vars?(subtermOF(lhs(e1), p))                              &
        mgu(sigma)(subtermOF(lhs(e1), p), lhs(e2))                    &
        t1 = ext(sigma)(rhs(e1))                                      &
        t2 = replaceTerm(ext(sigma)(rhs(e2)), ext(sigma)(lhs(e1)), p)
```

Knuth-Bendix Critical Pair Theorem

Specification

```
CP_Theorem: THEOREM
 FORALL E:
         local_confluent?(reduction?(E))
                      <=>
(FORALL t1, t2: CP?(E)(t1, t2) => joinable?(reduction?(E))(t1,t2))
```

Knuth-Bendix Critical Pair Theorem

A sketch of the formalisation

Let s be a term of divergence such that

$$\begin{tikzcd} & s \arrow[dl, "l_1 \rightarrow r_1"'] \arrow[dr, "l_2 \rightarrow r_2"] & \\ s_1 & & s_2 \end{tikzcd}$$

that is, there are positions $p_1, p_2 \in \texttt{positions?}(s)$, rules $l_1 \rightarrow r_1,\ l_2 \rightarrow r_2 \in \mathtt{E}$, and substitutions σ_1, σ_2, such that

$$s|_{p_1} = \sigma_1(l_1) \qquad \& \qquad s_1 = s[p_1 \leftarrow \sigma_1(r_1)]$$

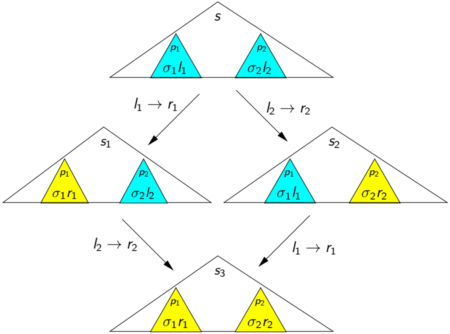
$$s|_{p_2} = \sigma_2(l_2) \qquad \& \qquad s_2 = s[p_2 \leftarrow \sigma_2(r_2)]$$

Knuth-Bendix Critical Pair Theorem

A sketch of the formalisation: Disjoint positions

p_1 and p_2 are in separate subtrees, i.e., p_1 and p_2 are parallel positions in s.

Case 1: Disjoint positions



Case 1: Disjoint positions

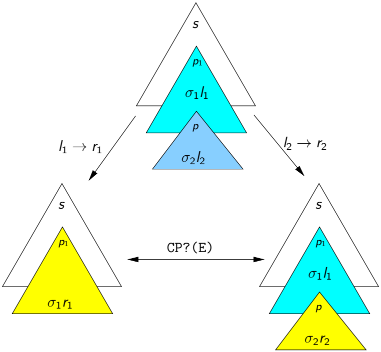
- Persistence
- Commutativity

Knuth-Bendix Critical Pair Theorem

A sketch of the formalisation: Critical overlap

$p \in$ `positions?`(l_1), $l_1|_p$ is not a variable and $\sigma_1(l_1|_p) = \sigma_2(l_2)$.

Case 2: Either $p_1 \leq p_2$ *or* $p_2 \leq p_1$ - $p_2 = p_1 p$



Knuth-Bendix Critical Pair Theorem

Case 2: The divergence corresponds to an instance of a critical pair $\langle t_1, t_2\rangle$

```
CP_lemma_aux1: LEMMA
 FORALL E, (e1 | member(e1, E)), (e2 | member(e2, E)), (p: position):
  positionsOF(lhs(e1))(p)                                            &
  NOT vars?(subtermOF(lhs(e1), p))                                   &
  ext(sg1)(subtermOF(lhs(e1), p)) = ext(sg2)(lhs(e2))
=>
  EXISTS t1, t2, delta:
  CP?(E)(t1, t2)                                                     &
  ext(delta)(t1) = ext(sg1)(rhs(e1))                                 &
  ext(delta)(t2) = replaceTerm(ext(sg2)(rhs(e2)), ext(sg1)(lhs(e1)), p)
```

Knuth-Bendix Critical Pair Theorem

In general the critical overlap case is proved in textbooks by assuming that the rewriting rules $l_1 \to r_1$ and $l_2 \to r_2$ are renamed such that $\mathtt{Vars}(l_1) \cap \mathtt{Vars}(l_2) = \emptyset$.

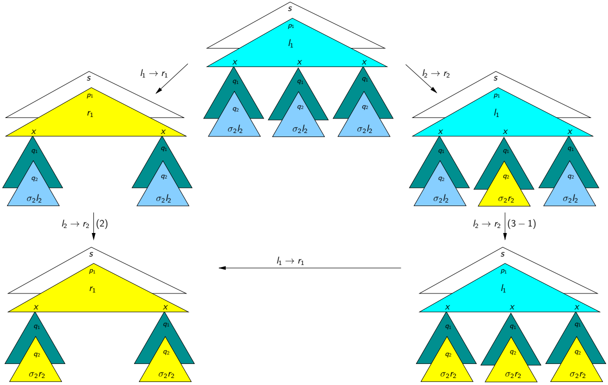
Case 2: Auxiliary properties

```
CP_lemma_aux1a: LEMMA
 FORALL E, (e1 | member(e1, E)), (e2 | member(e2, E)), (p: position):
  positionsOF(lhs(e1))(p)                                           &
  NOT vars?(subtermOF(lhs(e1), p))                                  &
  ext(sg1)(subtermOF(lhs(e1), p)) = ext(sg2)(lhs(e2)) )
=>
  EXISTS alpha, rho:
  disjoint?(Vars(lhs(e1)), Vars(ext(rho)(lhs(e2))))                 &
  ext(sg1)(subtermOF(lhs(e1), p)) = ext(comp(alpha, rho))(lhs(e2))
```

Knuth-Bendix Critical Pair Theorem

A sketch of the formalisation: Non-critical overlap

$p = q_1 q_2$, for q_2 possibly empty, such that q_1 is a position of variable in l_1 and $\sigma_2(l_2) = \sigma_1(l_1|_{q_1})|_{q_2}$.



Knuth-Bendix Critical Pair Theorem

Case 3: Auxiliary lemma

Let $\rightarrow$ be a relation compatible with the structure of terms, x be a variable, and σ_1 and σ_2 be substitutions such that:

$$\begin{array}{ccl} \sigma_1(x) & \rightarrow & \sigma_2(x) \text{ and} \\ \sigma_1(y) & = & \sigma_2(y), \text{ for all } y \neq x. \end{array}$$

Let t be an arbitrary term, and $p_1, \ldots, p_n \in \texttt{positions?}(t)$ be all the occurrences of x in t. Define $t_0 = \sigma_1(t)$ and $t_i = t_{i-1}[p_i \leftarrow \sigma_2(x)]$, for $1 \leq i \leq n$. Then $t_i \rightarrow^{n-i} \sigma_2(t)$, for $0 \leq i \leq n$. In particular, $\sigma_1(t) \rightarrow^n \sigma_2(t)$.

Case 3: Auxiliary constructors

```
replace_pos(t, s, (fssp:SPP(s)) ): RECURSIVE term =
    IF length(fssp) = 0 THEN  s
    ELSE replace_pos(t,replaceTerm(t, s, fssp(0)), rest(fssp)) ENDIF
  MEASURE length(fssp)

RSigma(R, sg1, sg2, x): bool = FORALL (y: (V)):
   IF y /= x THEN sg1(y) = sg2(y) ELSE R(sg1(x), sg2(x)) ENDIF
```

Knuth-Bendix Critical Pair Theorem

Case 3: The variable $l_1|_{q_1}$ can occur repeatedly in both sides of the rule $l_1 \to r_1$

```
CP_lemma_aux2: LEMMA
 FORALL R, t, x, sg1, sg2:
  LET Posv = Pos_var(t, x),
      seqv = set2seq(Posv) IN
  comp_cont?(R) & RSigma(R, sg1, sg2, x)
=>
  FORALL (i: below[length(seqv)]):
   RTC(R)(replace_pos(ext(sg2)(x),ext(sg1)(t), #(seqv(i))),ext(sg2)(t))
                                          &
                RTC(R)(ext(sg1)(t), ext(sg2)(t))
```

The PVS theory `orthogonality`

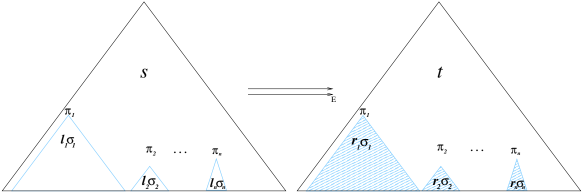
- The PVS theory `orthogonality` substantially enlarges the theory `trs` including several notions and formalisations related with the specification of orthogonal TRSs.

$\Rightarrow$ `orthogonality` includes a formalisation of the theorem of confluence of orthogonal TRSs according to:

  - use of the parallel reduction relation and
  - an inductive construction of terms of joinability for parallel divergences through the Parallel Moves Lemma.

Available: NASA LaRC PVS library or `trs.cic.unb.br`.

Parallel Rewriting



$s \rightrightarrows t$

$\rightrightarrows$(E)(s,t) : bool = $\exists$ (Π : SPP(t1), Γ : Seq[E], Σ : Seq[Subs]) : ...
t = replace_par_pos(s, Π, sigma_rhs(Σ,Γ))

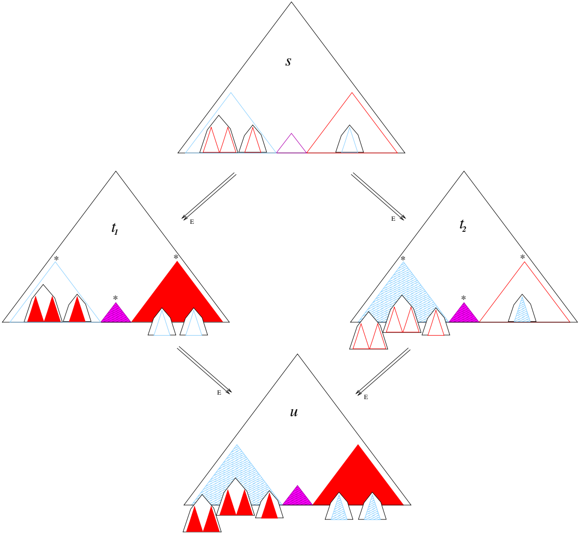
Theorem [Confluence of Orthogonal TRSs] *Orthogonality $\Rightarrow$ confluence*

One has to prove:

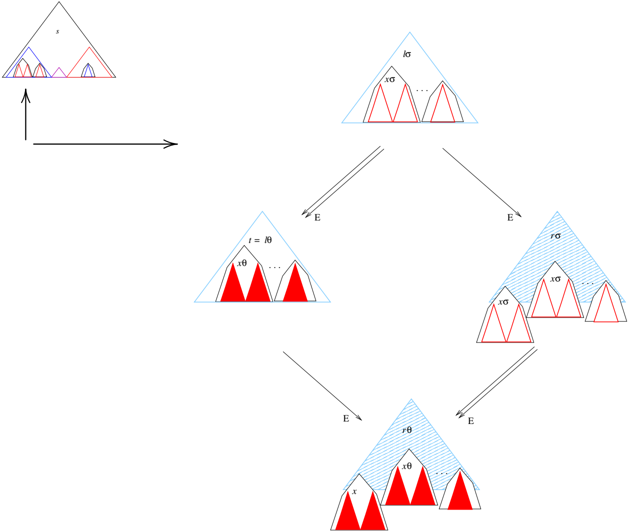
- **the diamond property ($\Diamond$P) for $\rightrightarrows$;**
- $\rightarrow \; \subseteq \; \rightrightarrows \; \subseteq \; \rightarrow^*$ implies $\rightrightarrows^* \; \equiv \; \rightarrow^*$;
- $\rightrightarrows$ confluent, implies $\rightarrow$ confluent.

$$\begin{tikzcd} & {} \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\ {} \arrow[dr, Rightarrow, dashed] & \Diamond P & {} \arrow[dl, Rightarrow, dashed] \\ & {} & \end{tikzcd} \quad \textit{implies} \quad \begin{tikzcd} & {} \arrow[dl, Rightarrow, "*"'] \arrow[dr, Rightarrow, "*"] & \\ {} \arrow[dr, Rightarrow, dashed, "*"'] & CR & {} \arrow[dl, Rightarrow, dashed, "*"] \\ & {} & \end{tikzcd} \quad \textit{implies} \quad \begin{tikzcd} & {} \arrow[dl, "*"'] \arrow[dr, "*"] & \\ {} \arrow[dr, dashed, "*"'] & CR & {} \arrow[dl, dashed, "*"] \\ & {} & \end{tikzcd}$$

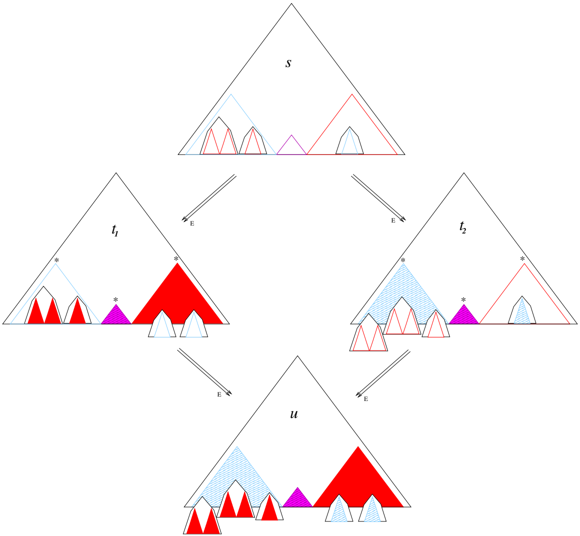
Orthogonal?(E) => diamond_property?(parallel_reduction?(E))



Building the joinability term: the Parallel Moves Lemma



Joinability requires synchronised applications of PML



Formalisation: Orthogonal_implies_confluent

Lemma (Specification of Orthogonality implies Confluence)

```
Orthogonal_implies_confluent:  LEMMA
     FORALL (E : Orthogonal) :
        confluent?(reduction?(E))
```

Formalisation: parallel_reduction_has_DP

Lemma (Specification of Orthogonality of $\rightarrow$ implies $\diamond P$ of $\rightrightarrows$)

```
parallel_reduction_has_DP: LEMMA

      Orthogonal?(E) =>

            diamond_property?(⇉(E))
```

Formalisation: divergence_in_Pos_Over

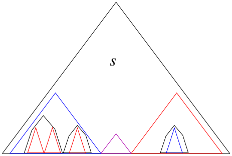
divergence_in_Pos_Over: LEMMA

$\rightrightarrows$(E)(s,t1,Π_1) $\wedge$ $\rightrightarrows$(E)(s,t2,Π_2) $\wedge$ $\pi \in$ Pos_Over(Π_1, Π_2)

=>

LET Π = complement_pos(π, Π_2) IN
$\exists$ ($(l, r) \in$ E , σ) :
subtermOF(s, π) = $l\sigma$ $\wedge$
subtermOF(t1, π) = $r\sigma$ $\wedge$
$\rightrightarrows$(E)(subtermOF(s, π),subtermOF(t2,π), Π)

Formalisation: subterm_joinability



subterm_joinability: LEMMA

Orthogonal?(E) $\wedge$ $\rightrightarrows$(E)(s,t1,Π_1) $\wedge$ $\rightrightarrows$(E)(s,t2,Π_2) $\wedge$
Π = Pos_Over(Π_1,Π_2) o Pos_Over(Π_2,Π_1) o Pos_Equal(Π_1,Π_2)

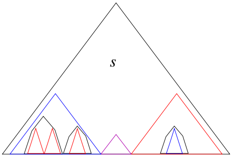
=>

$\forall i <| \Pi |$:

$\exists u_i$: $\rightrightarrows$(E)(subtermOF(t1, Π(i)), u_i) $\wedge$

$\rightrightarrows$(E)(subtermOF(t2, Π(i)), u_i)

Formalisation: subterms_joinability



subterms_joinability: LEMMA

Orthogonal?(E) $\wedge$ $\rightrightarrows$(E)(s,t1,Π_1) $\wedge$ $\rightrightarrows$(E)(s,t2,Π_2) $\wedge$
Π = Pos_Over(Π_1,Π_2) o Pos_Over(Π_2,Π_1) o Pos_Equal(Π_1,Π_2)

=>

$\exists$U : |U| = |Π| $\wedge$
$\forall$ i : $\rightrightarrows$(E)(subtermOF(t1, Π(i)), U(i)) $\wedge$
$\rightrightarrows$(E)(subtermOF(t2, Π(i)), U(i))

Conclusion and Future Work

- trs provides elegant formalisations close to textbook's and paper's proofs.

$$
\begin{tikzcd}
 & x \arrow[dl, "R^*"'] \arrow[dr, "R^*"] & \\
y \arrow[dr, "R^*"'] & \mathsf{CR} & z \arrow[dl, "R^*"] \\
 & \cdot &
\end{tikzcd}
$$

```
confluent?(R): bool = ∀( x, y, z):
    →*(R)(x,y) ∧ →*(R)(x,z)
        => ↓(R)(y,z)
```

$\Rightarrow$ First <u>straightforward</u> formalisation of Knuth-Bendix CP Th.
$\Rightarrow$ A formalisation of Rosen's confluence of orthogonal TRS's.

- Precise discrimination of notions and properties:
  - $\diamond$ property implies non termination.
  - proof's analogies fail: a development of parallel rewriting was necessary to formalise confluence of orthogonal TRS's.
- Clarity about adaptation of results in other contexts: confluence in *explicit substitutions* and *nominal rewriting*.

Conclusion and Future Work

- Applications to certify confluence of orthogonal specifications, variants of lambda calculus, nominal rewriting.
- Adaptation of the proof in Takahashi's style.
- Formalisations using other styles of proof. Van Oostrom's developments, for instance.
- Formalisations of termination: Joint work with Cesar Muñoz (NASA LaRC). PVS libraries `CCG` and `PVS0`.

Classical References

G. Huet.
A complete proof of correctness of the Knuth-Bendix completion algorithm.
Journal of Computer and Systems Sciences, 23, 1981.

D. E. Knuth and P. B. Bendix.
Computational Problems in Abstract Algebra, chapter Simple Words Problems in Universal Algebras, pages 263–297.
J. Leech, ed. Pergamon Press, Oxford, U. K., 1970.

M. H. A. Newman.
On theories with a combinatorial definition of "equivalence".
Ann. of Math., 43(2):223–243, 1942.

B. K. Rosen.
Tree-manipulating systems and church-rosser theorems.
J. of the ACM, 20(1):160–187, 1973.

Self References

A. L. Galdino and M. Ayala-Rincón.
A Formalization of Newman's and Yokouchi Lemmas in a Higher-Order Language.
J. of Formalized Reasoning, 1(1):39–50, 2008.

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

A. C. Rocha Oliveira, A. L. Galdino and M. Ayala-Rincón.
Confluence of Orthogonal Term Rewriting Systems in the Prototype Verification System
J. of Automated Reasoning, 58(2):231-251, 2017.