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 1st 2018

Nominal Unification

- Nominal Syntax
- Formalization of basic properties α-equivalence
- Formalization of Correctness of NU

Summary on NU

- Summary
- Future and work in progress on Nominal Unification

Dependency Pairs

- Termination of PVS0 Functions by TCC and SCP
- Dependency Pairs for PVS0 Functions
- Dependency Pairs for FPs versus termination TCCs

Summary on DP

- Summary
- Future work and work in progress on DP

Formalization of Nominal Unification

Formalization by Ana Cristina Rocha-Oliveira

What is unification?

- Goal: make two expressions equal.
- How: substitute variables by other expressions.
- Example: $f(x, a) \stackrel{?}{\approx}_? f(b, y)$.

  Solution: $\{x \mapsto b, y \mapsto a\}$.

What does unification work for?

- To instantiate a lemma in theorem proving.
- To proceed with a rewriting step (matching).
- To deduce critical pairs in a completion procedure.
- To infer types in programming languages.

First-order syntactic unification was formalized in PVS as part of `trs` theory [AGdMA14].

What is nominal good for?

- Deal with binders in an elegant way.
- Built in α-equivalence.
- First-order substitutions.
- Unification is decidable and polynomial.
- Frameworks based on nominal setting: α-Prolog, Fresh ML, Cαlm...

Unification modulo α-equivalence

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7} \sum_{i=0}^{5} (i - X)^i \;?\approx?\; \sum_{i=0}^{7} \sum_{k=0}^{5} (X - Y)^k$$

Unification modulo α-equivalence

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7} \sum_{i=0}^{5} (i - X)^i \;?\approx?\; \sum_{i=0}^{7} \sum_{k=0}^{5} (X - Y)^k$$

Unification modulo α-equivalence

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7} \sum_{i=0}^{5} (i - X)^i \;?\approx?\; \sum_{i=0}^{7} \sum_{k=0}^{5} (X - Y)^k$$

Unification modulo α-equivalence

Examples:

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7}\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{i=0}^{7}\sum_{k=0}^{5}(X - Y)^k$$

Solution: $[X \mapsto k][Y \mapsto i]$

$$\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5}(X - Y)^k \;\; ?$$

$$\lambda x \cdot (x\ M) \;{}_?{\approx}_?\; \lambda y \cdot (y\ M) \;\; ?$$

Unification modulo α-equivalence

Examples:

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7}\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{i=0}^{7}\sum_{k=0}^{5}(X - Y)^k$$

Solution: $[X \mapsto k][Y \mapsto i]$

$$\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5}(X - Y)^k \;\; ?$$

$$\lambda x \cdot (x\ M) \;{}_?{\approx}_?\; \lambda y \cdot (y\ M) \;\; ?$$

Unification modulo α-equivalence

Examples:

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

$$\sum_{k=0}^{7} \sum_{i=0}^{5} (i - X)^i \;{}_?{\approx}_?\; \sum_{i=0}^{7} \sum_{k=0}^{5} (X - Y)^k$$

Solution: $[X \mapsto k][Y \mapsto i]$

$$\sum_{i=0}^{5} (i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5} (X - Y)^k \;\; ?$$

$$\lambda x \cdot (x\ M) \;{}_?{\approx}_?\; \lambda y \cdot (y\ M) \;\; ?$$

Unification modulo α-equivalence

Examples:

$$x^2 + 3x + 1 \approx_\alpha y^2 + 3y + 1$$

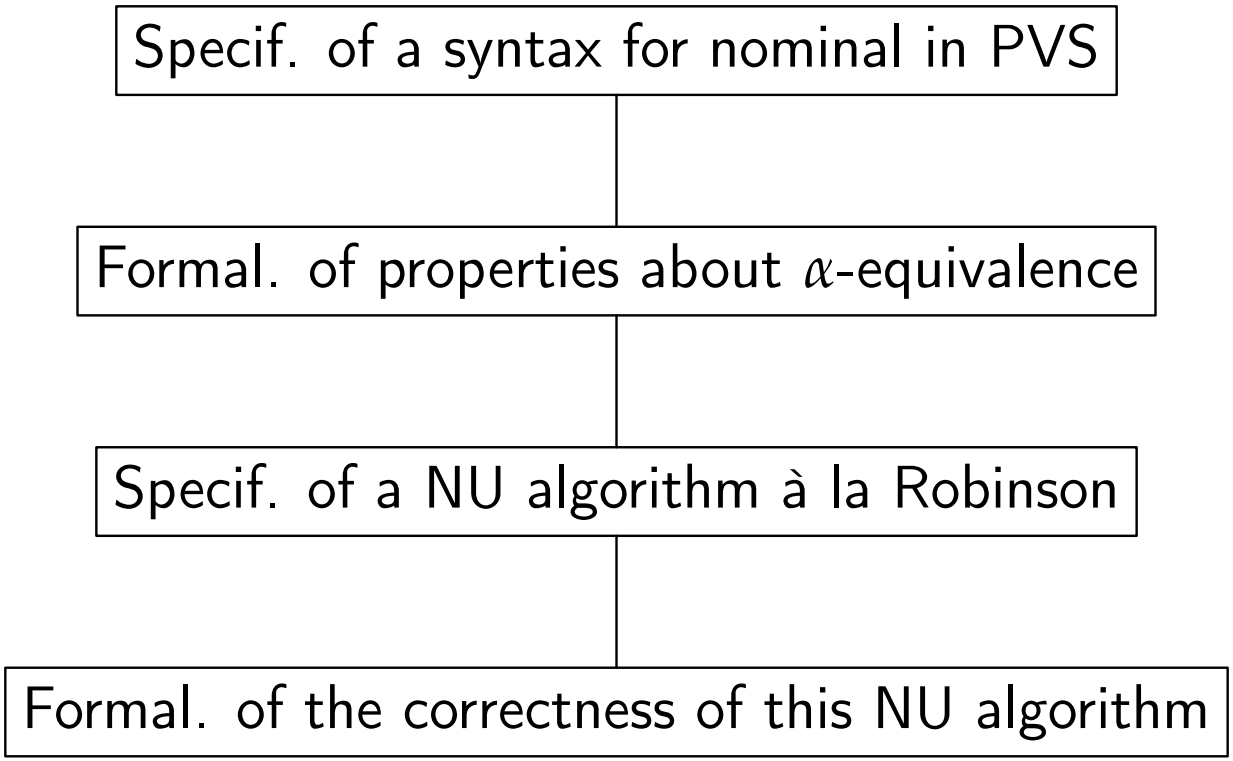
$$\sum_{k=0}^{7} \sum_{i=0}^{5} (i - X)^i \;{}_?{\approx}_?\; \sum_{i=0}^{7} \sum_{k=0}^{5} (X - Y)^k$$

Solution: $[X \mapsto k][Y \mapsto i]$

$$\sum_{i=0}^{5} (i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5} (X - Y)^k \;\; ?$$

$$\lambda x \cdot (x\ M) \;{}_?{\approx}_?\; \lambda y \cdot (y\ M) \;\; ?$$

PVS Development



Nominal terms - Data structure

$$s, t ::= \bar{a} \mid \pi \cdot X \mid () \mid (s, t) \mid [a]s \mid f\ t$$

```
term[atom:TYPE+, perm:TYPE+, variable:TYPE+,
symbol:TYPE+]:
   DATATYPE
     BEGIN
         at (a:  atom):  atom?
         * (p:  perm, V: variable):  susp?
         unit:  unit?
         pair (term1:  term, term2:  term):  pair?
         abs (abstr:  atom, body:  term):  abs?
         app (sym:  symbol, arg:  term):  app?
     END term
```

α-equivalence: deduction rules

$$\frac{}{\nabla \vdash \bar{a} \approx_\alpha \bar{a}}\ (\approx_\alpha \mathtt{a}) \qquad \frac{ds(\pi, \pi')\#X \subseteq \nabla}{\nabla \vdash \pi \cdot X \approx_\alpha \pi' \cdot X}\ (\approx_\alpha \mathtt{X})$$

$$\frac{}{\nabla \vdash () \approx_\alpha ()}\ (\approx_\alpha ()) \qquad \frac{\nabla \vdash s \approx_\alpha t}{\nabla \vdash f\ s \approx_\alpha f\ t}\ (\approx_\alpha \mathtt{f})$$

$$\frac{\nabla \vdash s_1 \approx_\alpha t_1 \quad \nabla \vdash s_2 \approx_\alpha t_2}{\nabla \vdash (s_1, s_2) \approx_\alpha (t_1, t_2)}\ (\approx_\alpha \mathtt{pair})$$

$$\frac{\nabla \vdash s \approx_\alpha t}{\nabla \vdash [a]s \approx_\alpha [a]t}\ (\approx_\alpha \mathtt{absa}) \qquad \frac{\nabla \vdash s \approx_\alpha (a\ b) \cdot t \quad \nabla \vdash a\#t}{\nabla \vdash [a]s \approx_\alpha [b]t}\ (\approx_\alpha \mathtt{absb})$$

where $\nabla = \{(a_1\#X_1), (a_2\#X_2), \ldots, (a_n\#X_n)\}$.

Freshness: deduction rules

$$\frac{}{\nabla \vdash a\#\bar{b}}\ (\#\mathtt{ab}) \qquad \frac{(\pi^{-1}(a), X) \in \nabla}{\nabla \vdash a\#\pi \cdot X}\ (\#\mathtt{X})$$

$$\frac{}{\nabla \vdash a\#()}\ (\#\mathtt{unit}) \qquad \frac{\nabla \vdash a\#s}{\nabla \vdash a\#f\ s}\ (\#\mathtt{f})$$

$$\frac{\nabla \vdash a\#s_1 \qquad \nabla \vdash a\#s_2}{\nabla \vdash a\#(s_1, s_2)}\ (\#\mathtt{pair})$$

$$\frac{}{\nabla \vdash a\#[a]s}\ (\#\mathtt{absa}) \qquad \frac{\nabla \vdash a\#s}{\nabla \vdash a\#[b]s}\ (\#\mathtt{absb})$$

α-equivalence is indeed an equivalence relation

Lemma (`alpha_reflexive`)

$$\Delta \vdash t \approx_\alpha t$$

Lemma (`alpha_transitive`)

$$\Delta \vdash t_1 \approx_\alpha t_2 \texttt{ AND } \Delta \vdash t_2 \approx_\alpha t_3 \texttt{ => } \Delta \vdash t_1 \approx_\alpha t_3$$

Lemma (`alpha_symmetric`)

$$\Delta \vdash t \approx_\alpha s \quad = \quad \Delta \vdash s \approx_\alpha t$$

Proof of transitivity

Form. of trans. in Isabelle/HOL: difficult induction scheme [Urb04]

Alternative weak equivalence to ease the proof [Urb10]

Form. of trans. in PVS: simpler proof and free of weak equiv.

Proof of transitivity in Isabelle/HOL [Urb04]

lemma big: "$\forall t1\ t2\ t3.\ (n = depth\ t1) \rightarrow$
$((nabla \vdash t1 \approx t2) \rightarrow (nabla \vdash t2 \approx t1)) \wedge$
$(\forall pi.(nabla \vdash t1 \approx t2) \rightarrow (nabla \vdash swap\ pi\ t1 \approx swap\ pi\ t2)) \wedge$
$((nabla \vdash t1 \approx t2) \wedge (nabla \vdash t2 \approx t3) \rightarrow (nabla \vdash t1 \approx t3))$"

The symbol $\approx$ in Isabelle/HOL refers to $\approx_\alpha$ in our notation.

Weak-equivalence by [Urb10]

$$s \sim t$$

$$\frac{}{\bar{a} \sim \bar{a}}\ (\sim\mathtt{a}) \qquad \frac{ds(\pi, \pi') = \varnothing}{\pi \cdot X \sim \pi' \cdot X}\ (\sim\mathtt{X}) \qquad \frac{}{() \sim ()}\ (\sim\mathtt{()})$$

$$\frac{s_1 \sim t_1 \quad s_2 \sim t_2}{(s_1, s_2) \sim (t_1, t_2)}\ (\sim\mathtt{pair}) \qquad \frac{s \sim t}{[a]s \sim [a]t}\ (\sim\mathtt{absa}) \qquad \frac{s \sim t}{f\ s \sim f\ t}\ (\sim\mathtt{f})$$

$$(\Delta \vdash t_1 \approx t_2 \wedge t_2 \sim t_3) \Rightarrow \Delta \vdash t_1 \approx t_3.$$

PVS formalization is free of weak equivalence and has simpler induction schemes (files available at `trs.cic.unb.br`).

Nominal unification by Urban $- \exists \Delta, \sigma : \ \Delta \vdash s\sigma \approx_\alpha t\sigma$?

$$
\begin{array}{rcll}
a\#\bar{b}, Pr & \Longrightarrow & Pr & \\
a\#\pi \cdot X, Pr & \overset{\{(\pi^{-\mathbf{1}} \cdot a, X)\}}{\Longrightarrow} & Pr & \\
a\#(\,), Pr & \Longrightarrow & Pr & \\
a\#(s_1, s_2), Pr & \Longrightarrow & a\#s_1, a\#s_2, Pr & \\
a\#[b]s, Pr & \Longrightarrow & a\#s, Pr & \\
a\#[a]s, Pr & \Longrightarrow & Pr & \\
a\#f\ s, Pr & \Longrightarrow & a\#s, Pr & \\
\bar{a} \ {}_?{\approx}_? \ \bar{a}, Pr & \Longrightarrow & Pr & \\
\pi \cdot X \ {}_?{\approx}_? \ \pi' \cdot X, Pr & \Longrightarrow & ds(\pi, \pi')\#X, Pr & \\
(\,) \ {}_?{\approx}_? \ (\,), Pr & \Longrightarrow & Pr & \\
(s_1, s_2) \ {}_?{\approx}_? \ (t_1, t_2), Pr & \Longrightarrow & s_1 \ {}_?{\approx}_? \ t_1, s_2 \ {}_?{\approx}_? \ t_2, Pr & \\
[a]s \ {}_?{\approx}_? \ [a]t, Pr & \Longrightarrow & s \ {}_?{\approx}_? \ t, Pr & \\
[b]s \ {}_?{\approx}_? \ [a]t, Pr & \Longrightarrow & (a\ b) \cdot s \ {}_?{\approx}_? \ t, a\#s, Pr & \\
f\ s \ {}_?{\approx}_? \ f\ t, Pr & \Longrightarrow & s \ {}_?{\approx}_? \ t, Pr & \\
\pi \cdot X \ {}_?{\approx}_? \ u, Pr & \overset{[X \mapsto \pi^{-\mathbf{1}} \cdot u]}{\Longrightarrow} & Pr[X \mapsto \pi^{-1} \cdot u] & (X \notin Vars(u)) \\
u \ {}_?{\approx}_? \ \pi \cdot X, Pr & \overset{[X \mapsto \pi^{-\mathbf{1}} \cdot u]}{\Longrightarrow} & Pr[X \mapsto \pi^{-1} \cdot u] & (X \notin Vars(u))
\end{array}
$$

Nominal unification by Urban

Examples:

- $\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5}(X - Y)^k$
  $\Longrightarrow (k - (i\,k)\cdot X)^k \;{}_?{\approx}_?\; (X - Y)^k, k\#(i - X)^i$
  $\Longrightarrow k \;{}_?{\approx}_?\; X, (i\,k)\cdot X \;{}_?{\approx}_?\; Y, k \;{}_?{\approx}_?\; k, k\#(i - X)^i$
  $\overset{[X\mapsto k]}{\Longrightarrow} i \;{}_?{\approx}_?\; Y, k\#(i - k)^i$

  problem!

- $\lambda x\cdot(x\ M) \;{}_?{\approx}_?\; \lambda y\cdot(y\ M)$
  $\Longrightarrow (y\ (x\,y)\cdot M) \;{}_?{\approx}_?\; (y\ M), y\#(x\ M)$
  $\Longrightarrow y \;{}_?{\approx}_?\; y, (x\,y)\cdot M \;{}_?{\approx}_?\; M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M$

  Solution: $\langle\{x\#M, y\#M\}, Id\rangle$

Nominal unification by Urban

Examples:

- $\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5}(X - Y)^k$
  $\Longrightarrow (k - (i\,k)\cdot X)^k \;{}_?{\approx}_?\; (X - Y)^k, k\#(i - X)^i$
  $\Longrightarrow k \;{}_?{\approx}_?\; X, (i\,k)\cdot X \;{}_?{\approx}_?\; Y, k \;{}_?{\approx}_?\; k, k\#(i - X)^i$
  $\overset{[X\mapsto k]}{\Longrightarrow} i \;{}_?{\approx}_?\; Y, k\#(i - k)^i$

  problem!

- $\lambda x\cdot(x\ M) \;{}_?{\approx}_?\; \lambda y\cdot(y\ M)$
  $\Longrightarrow (y\ (x\,y)\cdot M) \;{}_?{\approx}_?\; (y\ M), y\#(x\ M)$
  $\Longrightarrow y \;{}_?{\approx}_?\; y, (x\,y)\cdot M \;{}_?{\approx}_?\; M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M$

  Solution: $\langle\{x\#M, y\#M\}, Id\rangle$

Nominal unification by Urban

Examples:

- $\sum_{i=0}^{5}(i - X)^i \;{}_?{\approx}_?\; \sum_{k=0}^{5}(X - Y)^k$
  $\Longrightarrow (k - (i\,k)\cdot X)^k \;{}_?{\approx}_?\; (X - Y)^k, k\#(i - X)^i$
  $\Longrightarrow k \;{}_?{\approx}_?\; X, (i\,k)\cdot X \;{}_?{\approx}_?\; Y, k \;{}_?{\approx}_?\; k, k\#(i - X)^i$
  $\overset{[X\mapsto k]}{\Longrightarrow} i \;{}_?{\approx}_?\; Y, k\#(i - k)^i$

  problem!

- $\lambda x\cdot(x\ M) \;{}_?{\approx}_?\; \lambda y\cdot(y\ M)$
  $\Longrightarrow (y\ (x\,y)\cdot M) \;{}_?{\approx}_?\; (y\ M), y\#(x\ M)$
  $\Longrightarrow y \;{}_?{\approx}_?\; y, (x\,y)\cdot M \;{}_?{\approx}_?\; M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M$

  Solution: $\langle\{x\#M, y\#M\}, Id\rangle$

Nominal unification by Urban

Examples:

- $\sum_{i=0}^{5}(i - X)^i \;?\approx?\; \sum_{k=0}^{5}(X - Y)^k$
  $\Longrightarrow (k - (i\,k)\cdot X)^k \;?\approx?\; (X - Y)^k, k\#(i - X)^i$
  $\Longrightarrow k \;?\approx?\; X, (i\,k)\cdot X \;?\approx?\; Y, k \;?\approx?\; k, k\#(i - X)^i$
  $\overset{[X\mapsto k]}{\Longrightarrow} i \;?\approx?\; Y, k\#(i - k)^i$

  problem!

- $\lambda x\cdot(x\ M) \;?\approx?\; \lambda y\cdot(y\ M)$
  $\Longrightarrow (y\ (x\,y)\cdot M) \;?\approx?\; (y\ M), y\#(x\ M)$
  $\Longrightarrow y \;?\approx?\; y, (x\,y)\cdot M \;?\approx?\; M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M, y\#(x\ M)$
  $\Longrightarrow x\#M, y\#M$

  Solution: $\langle\{x\#M, y\#M\}, Id\rangle$

Freshness problem: $\exists \Delta : \Delta \vdash a\#s$?

$$
\begin{array}{l}
\texttt{fresh?}(a,t) : \texttt{RECURSIVE}\,[\texttt{fresh\_context}, \texttt{bool}] = \\
\quad \texttt{CASES}\ t\ \texttt{OF} \\
\qquad \bar{b} : \langle \emptyset,\ a \neq b \rangle, \\
\qquad \pi \cdot X :\ \langle \{(\pi^{-1} \cdot a\#X)\},\ \texttt{True} \rangle, \\
\qquad () :\ \langle \emptyset,\ \texttt{True} \rangle, \\
\\
(t_1, t_2) :\ \texttt{LET}\ \langle \Delta_1, \mathtt{b}_1 \rangle = \texttt{fresh?}(a, t_1),\ \langle \Delta_2, \mathtt{b}_2 \rangle = \texttt{fresh?}(a, t_2)\ \texttt{IN} \\
\qquad\qquad\qquad \langle \Delta_1 \Delta_2,\ \mathtt{b}_1 \wedge \mathtt{b}_2 \rangle \\
\qquad [b]\hat{t} :\ \texttt{IF}\ a = b\ \texttt{THEN}\ \langle \emptyset, \texttt{True} \rangle \\
\qquad\qquad\qquad \texttt{ELSE}\ \texttt{fresh?}(a, \hat{t})\ \texttt{ENDIF}, \\
\qquad f\ \hat{t} :\ \texttt{fresh?}(a,\ \hat{t}) \\
\quad \texttt{ENDCASES} \\
\quad \texttt{MEASURE}\ t\ \texttt{BY}\ << \\
\\
\hline
\end{array}
$$

Notation: $\texttt{fresh?}(a,t) = \langle a\#t \rangle_{sol}$ and $\langle \Delta\sigma \rangle_{sol} = \bigcup_{(a\#X)\in\Delta} \langle a\#X\sigma \rangle_{sol}$.

Freshness - lemmas

Lemma (`fresh?_is_sound`)

$$\langle a\#t \rangle_{sol} = \langle \Delta, \texttt{True} \rangle \quad \texttt{=>} \quad \Delta \vdash a\#t.$$

Lemma (`fresh?_complete`)

$$\texttt{LET } \langle \Delta, \texttt{b} \rangle = \langle a\#t \rangle_{sol} \texttt{ IN}$$
$$\Gamma \vdash a\#t \quad \texttt{=>} \quad \texttt{b} \quad \texttt{AND } \Gamma \vdash \Delta$$

Nominal Unification Algorithm - PVS

$\texttt{unify}(t, s) : \texttt{RECURSIVE [fresh\_context, Subs\_unif}(t, s)\texttt{, bool]}$
$=$
$\quad \texttt{IF } s = \pi_s \cdot X_s \texttt{ AND } X_s \notin \mathit{Vars}(t)$
$\quad\quad \texttt{THEN} \langle \emptyset, [X_s \mapsto \pi_s^{-1} \bullet t],\ \mathit{True} \rangle$
$\quad\quad \texttt{ELSE}$
$\quad\quad\quad \texttt{CASES } (t, s) \texttt{ OF}$
$\quad\quad\quad\quad (\pi_t \cdot X, \pi_s \cdot X) : \langle ds(\pi_t, \pi_s)\#X,\ \mathit{Id},\ \mathit{True} \rangle,$
$\quad\quad\quad\quad (\pi_t \cdot X_t,\ s) :\ \texttt{IF } X_t \notin \mathit{Vars}(s)$
$\quad\quad\quad\quad\quad\quad\quad\quad \texttt{THEN } \langle \emptyset, [X_t \mapsto \pi_t^{-1} \cdot s],\ \mathit{True} \rangle$
$\quad\quad\quad\quad\quad\quad\quad\quad \texttt{ELSE } \langle \emptyset,\ \mathit{Id},\ \mathit{False} \rangle$
$\quad\quad\quad\quad\quad\quad\quad \texttt{ENDIF},$
$\quad\quad\quad\quad (\bar{a}, \bar{a}) :\ \langle \emptyset,\ \mathit{Id},\ \mathit{True} \rangle,$
$\quad\quad\quad\quad ((),\ ()) : \langle \emptyset,\ \mathit{Id},\ \mathit{True} \rangle,$
$\quad\quad\quad\quad (f\ \hat{t},\ f\ \hat{s}) : \texttt{unify}(\hat{t},\ \hat{s}),$

Nominal Unification Algorithm - PVS

$$
\begin{array}{l}
((t_1, t_2),\ (s_1, s_2)) :\ \texttt{LET}\ \langle \Delta_1,\ \sigma_1,\ \mathtt{b}_1\rangle = \mathtt{unify}(t_1, s_1), \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \Delta_2,\ \sigma_2,\ \mathtt{b}_2\rangle = \mathtt{unify}(t_2\sigma_1,\ s_2\sigma_1), \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \Delta_3,\ \mathtt{b}_3\rangle = \langle \Delta_1\sigma_2\rangle_{sol}\ \texttt{IN} \\
\qquad\qquad\qquad\qquad\qquad\ \texttt{IF}\ \mathtt{b}_1 \wedge \mathtt{b}_2 \wedge \mathtt{b}_3 \\
\qquad\qquad\qquad\qquad\qquad\qquad \texttt{THEN}\ \langle \Delta_2\Delta_3,\ \sigma_1 \circ \sigma_2,\ \mathit{True}\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \texttt{ELSE}\ \langle \emptyset,\ \mathit{Id},\ \mathit{False}\rangle\ \texttt{ENDIF}, \\
([a]\hat{t}, [b]\hat{s}) :\ \texttt{IF}\ a = b\ \texttt{THEN}\ \mathtt{unify}(\hat{t},\ \hat{s}) \\
\qquad\qquad\qquad\qquad\qquad\ \texttt{ELSE LET}\ \langle \Delta_1,\ \sigma,\ \mathtt{b}_1\rangle = \mathtt{unify}(\hat{t},\ (a, b)\cdot\hat{s}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \Delta_2,\ \mathtt{b}_2\rangle = \langle a\#\hat{s}\sigma\rangle_{sol}\ \texttt{IN} \\
\qquad\qquad\qquad\qquad\qquad\ \texttt{IF}\ \mathtt{b}_1 \wedge \mathtt{b}_2\ \texttt{THEN}\ \langle \Delta_1\Delta_2,\ \sigma,\ \mathit{True}\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \texttt{ELSE}\ \langle \emptyset,\ \mathit{Id},\ \mathit{False}\rangle,\ \texttt{ENDIF}, \\
\qquad\qquad\texttt{ELSE} :\ \langle \emptyset,\ \mathit{Id},\ \mathit{False}\rangle \\
\texttt{ENDCASES ENDIF} \\
\texttt{MEASURE}\ \mathit{lex}(|\mathit{Vars}(t, s)|,\ \mathit{depth}(t))
\end{array}
$$

Unification - lemmas

Lemma (`unify_sound`)

$$\texttt{LET } \langle \Delta,\ \sigma,\ \texttt{b} \rangle\ =\ \texttt{unify}(\texttt{t}, \texttt{s})\ \texttt{IN}$$
$$\texttt{b} \Rightarrow \Delta \vdash \texttt{t}\sigma \approx_\alpha \texttt{s}\sigma$$

Theorem (`unify_complete`)

$$\texttt{LET } \langle \Delta,\ \sigma,\ \texttt{b} \rangle\ =\ \texttt{unify}(\texttt{t}, \texttt{s})\ \texttt{IN}$$
$$\Gamma \vdash \texttt{t}\gamma \approx_\alpha \texttt{s}\gamma \Rightarrow\ \ \texttt{b}\ \ \texttt{AND}\ (\Delta,\ \sigma) \leq (\Gamma,\ \gamma)$$

Proofs by induction on $\mathit{lex}(|\mathit{Vars}(t, s)|,\ \mathit{depth}(t))$.

Summary

- The formalized algorithm avoids carrying freshness contexts as parameters and it is closer to the first-order unification algorithm à la Robinson [AFRO16].
- Separating freshness problems from equational problems highlights their indenpendence.
- The present proof of transitivity of α-equivalence is straightforward and free of a weak-equivalence relation.

Future and work in progress on NU

- Compare and adjust the present algorithm to efficient versions as in [CF08, LV10].
- Build a PVS theory for nominal rewriting (nominal matching is necessary for the rewrite step and unification identifies overlaps between rules): confluence properties do not hold as in the classic TRS approach [AFGRO16].
- Develop an intersection type system for nominal terms [ARFROV18].
- Study nominal unification modulo. Progress includes the introduction of nominal equational-check modulo A, C, AC theories, and nominal unification and matching modulo C [ARCSFNS17], [AdCSFN18], [AFN18].


Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, and Daniele Nantes-Sobrinho.
Nominal C-Unification.
In *Logic-Based Program Synthesis and Transformation - 27th International Symposium, LOPSTR 2017, Namur, Belgium, October 10-12, 2017, Revised Selected Papers*, volume 10855 of *Lecture Notes in Computer Science*, pages 235–251. Springer, 2018.


M. Ayala-Rincón, M. Fernández, M. J. Gabbay, and A. C. Rocha-Oliveira.
Checking overlaps of nominal rewriting rules.
Electr. Notes Theor. Comput. Sci., 323:39–56, 2016.


Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho.
Fixed-point constraints for nominal equational unification.
In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, volume 108 of *LIPIcs*, pages 7:1–7:16, 2018.


M. Ayala-Rincón, M. Fernández, and A. C. Rocha-Oliveira.
Completeness in PVS of a nominal unification algorithm.
Electr. Notes Theor. Comput. Sci., 323:57–74, 2016.


A. B. Avelar, A. L. Galdino, F. L. C. de Moura, and M. Ayala-Rincón.
First-order unification in the PVS proof assistant.
Logic Journal of the IGPL, 22(5):758–789, 2014.


M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho.
On Solving Nominal Fixpoint Equations.
In *Proc. of the 11th Int. Symp. on Frontiers of Combining Systems (FroCoS)*, volume 10483 of *Lecture Notes in Computer Science*, pages 209–226. Springer, 2017.

M. Ayala-Rincón, M. Fernández, A.C. Rocha-Oliveira, and D. L. Ventura.
Nominal essential intersection types.
Theor. Comput. Sci., 737:62–80, 2018.

C. Calvès and M. Fernández.
A Polynomial Nominal Unification Algorithm.
Theor. Comput. Sci., 403(2-3):285–306, 2008.

J. Levy and M. Villaret.
An efficient nominal unification algorithm.
In *Proceedings of the 21st RTA*, pages 209–226, 2010.

C. Urban.
Formalisation of Nominal Unification in Isabelle/HOL.
Technical report, TU Munich, 2004.
http://www4.in.tum.de/∼urbanc/Unification, last visited April 2015.

C. Urban.
Nominal Unification Revisited.
In *Proceedings 24th UNIF.*, volume 42 of *EPTCS*, pages 1–11, 2010.

Termination by Dependency Pairs of PVS0 Programs

Formalization in progress by Ariane Alves 

Grammar

$$expr ::= cnst | vr | op1(expr) | op2(expr, expr) | ite(expr, expr, expr) | rec(expr)$$

Termination by Dependency Pairs of PVS0 Programs

FP for Ackermann

```
ite(m=0,
      n+1,
      ite(n=0,
            ack(m-1,1),
            ack(m-1,ack(m,n-1))))
```

Expression PVS0 for Ackermann

```
ite(op1(0,vr),
      op1(2,vr),
      ite(op1(1,vr),
            rec(op1(3,vr)),
            rec(op2(0,
                  vr,
                  rec(op1(4,vr))))
```

```
boole(b:bool) : Val = IF b THEN
true_val ELSE false_val ENDIF

eop1: list[[Val->Val]] =
(: LAMBDA(m,n:nat): boole(m=0),
   LAMBDA(m,n:nat): boole(n=0),
   LAMBDA(m,n:nat): (n+1,0),
   LAMBDA(m,n:nat): IF m /= 0 THEN
              (m-1,1) ELSE (m,n) ENDIF,
   LAMBDA(m,n:nat): IF n /= 0 THEN
              (m,n-1) ELSE (m,n) ENDIF,
   LAMBDA(m,n:nat): false_val :)

eop2: list[[[Val,Val]->Val]] =
(: LAMBDA(v1,v2: Val):IF v1`1 > 0 THEN
 (v1`1-1,v2`1) ELSE false_val ENDIF :)
```

Functional Programs in PVS0 - Semantics

Given a PVS0 Program $pvs0 = (O_1, O_2, \bot, expr)$, and an input i, there are a predicate ε, and a function χ for semantic evaluation.

1. $\varepsilon(O_1, O_2, \bot, expr)(expr', i, o)$, where
   - $expr'$ is a subexpression of $expr$ being evaluated.
   - o is the output
2. $\chi(O_1, O_2, \bot, expr)(expr', i, n)$, where
   - n is the maximum allowed length of nested recursive calls.
   - If this limited is reached, the function returns $\diamondsuit$.
   - Otherwise it returns the output of evaluation.

Termination by Semantic Evaluation

- $T_\varepsilon(pvs0, i) := \exists\, o \in Val : \varepsilon(pvs0)(pvs0_e, i, o)$
- $T_\chi(pvs0, i) := \exists\, n \in \mathbb{N} : \chi(pvs0)(pvs0_e, i, n) \neq \diamondsuit$
- $T_\varepsilon(pvs0) := \forall i \in Val : T_\varepsilon(pvs0, i)$
- $T_\chi(pvs0) := \forall i \in \mathbb{N} : T_\chi(pvs0, i)$

TCC Termination for PVS0 Functions

TCC Termination

For tracing the right call, we use positions given as paths (Path : TYPE = list[nat]):

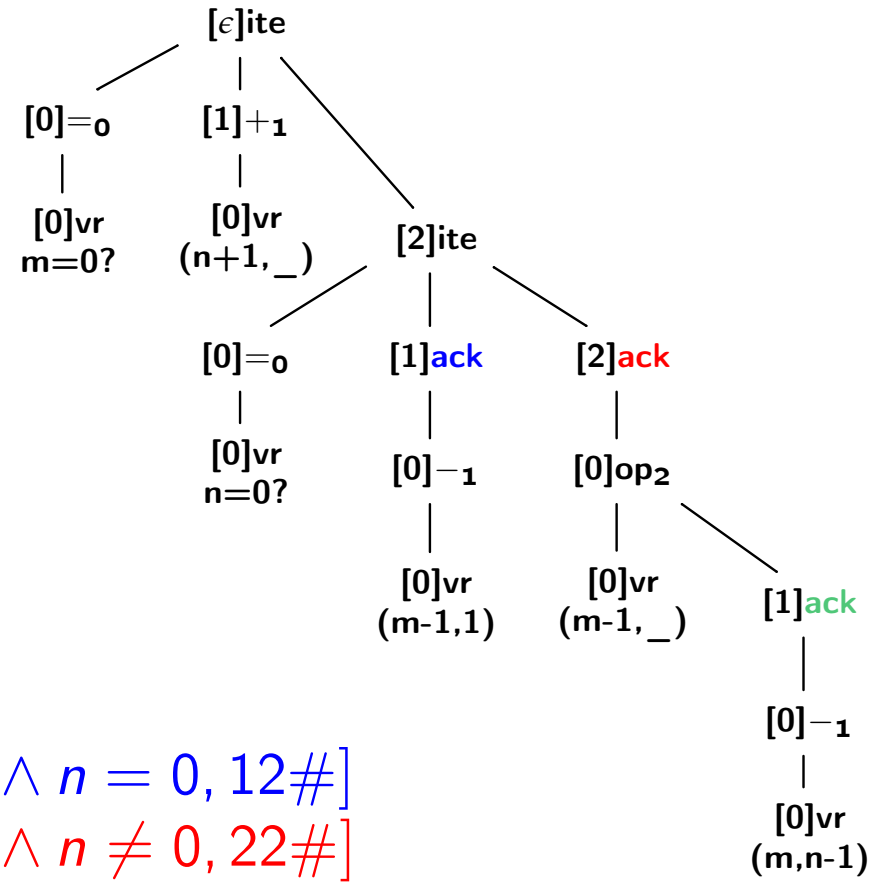
$$PVS0Expr\_CC : TYPE = [\# \quad rec\_expr : (rec?),\ cnds : Conditions,\ path : Path\ \#]$$

A PVS0 program *pvs*0 is TCC terminating if there exist a well-founded ordering *lt* and a measure μ, such that for all *cc* and values v_{in}, v_{out}:

$$\forall (v_{in}, v_{out}, cc) : (cond\_eval(pvs0)(cc'\,cnds, v_{in}) \wedge \varepsilon(pvs0)(get\_arg(cc'\,rec\_expr), v_{in}, v_{out}) \Rightarrow lt(\mu(v_{out}), \mu(v_{in})))$$

Paths and Conditions - Ackermann

```
ack(m,n) = ite(m=0,
               n+1,
               ite(n=0,
                   ack(m-1,1),
                   ack(m-1,ack(m,n-1))))
```



- Cond: $m \neq 0 \wedge n = 0$; Path: 12
- Cond: $m \neq 0 \wedge n \neq 0$; Path: 22
- Cond: $m \neq 0 \wedge n \neq 0$; Path: 1022

Ackermann's CCs

$$cc1 : [\#ack, m \neq 0 \wedge n = 0, 12\#]$$
$$cc2 : [\#ack, m \neq 0 \wedge n \neq 0, 22\#]$$
$$cc3 : [\#ack, m \neq 0 \wedge n \neq 0, 1022\#]$$

Size Change Principle

Size Change Principle - SCP

If every possibly infinite computation leads to an infinite decreasing over a well-founded order, then the program is terminating.

- No longer a local analisys -

Infinite Computation

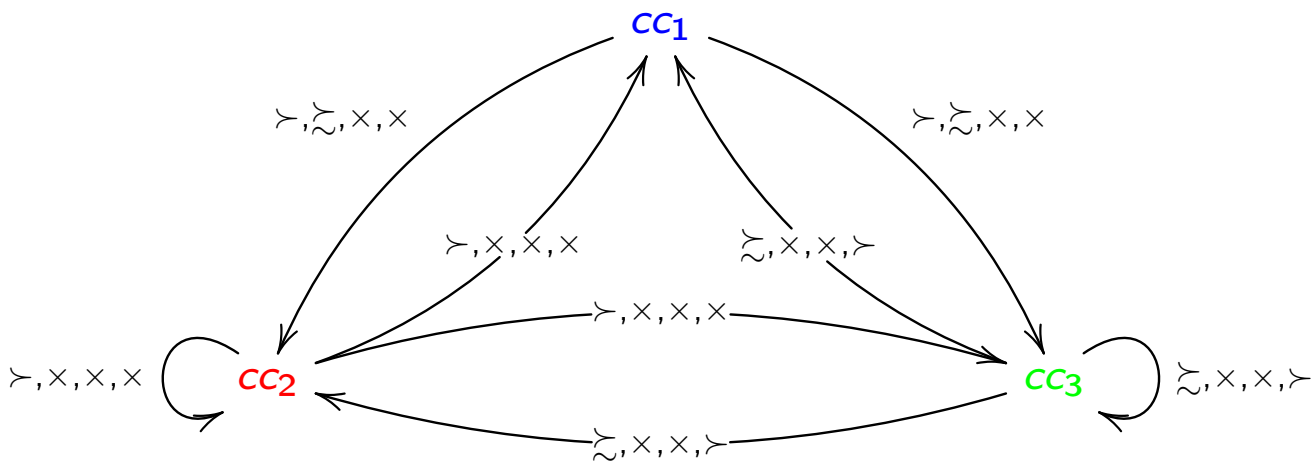
infinite_seq_ccs(pvs0, ccs, vals) =
$\forall$(i): cond_eval(pvs0)(ccs(i)‘cnds, vals(i)) $\wedge$
ε(pvs0)(ccs(i)‘actuals,vals(i),vals(i+1))

SCP Termination

$\forall$(ccs, vals): $\neg$ infinite_seq_ccs(pvs0)(ccs, vals)

Calling Context Graph

- Every CC is a vertex;
- There exists an edge from cc_i to cc_j if $cc_i cc_j$ is a *sequence*;
- Measure functions are associated to the formal and actual parameters of each CC;
- Results of comparisons are stored in each edge;
- Verify if each possible circuit in the graph have a combination of measures with strict decreasement.



Dependency Pairs

Given a TRS $\mathcal{R} = (C, D, R)$,
$DP(\mathcal{R}) = \{\langle L, T\rangle | l \to r \in R,$ and $r \triangleright t$ head by a symbol in $D\}$,
where for every $f \in D$, F is its *tuple symbol*.

TRS for Ackermann

1) $a(0, y) \to s(y)$
2) $a(s(x), 0) \to a(x, s(0))$
3) $a(s(x), s(y)) \to a(x, a(s(x), y))$

DPs for Ackermann

- $\langle A(s(x), 0), A(x, s(0))\rangle$
- $\langle A(s(x), s(y)), A(x, a(s(x), y))\rangle$
- $\langle A(s(x), s(y)), A(s(x), y)\rangle$

Termination by Dependency Pairs

Dependency Chain

A sequence of DPs $\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, \langle s_3, t_3\rangle \cdots$ is a *Dependency Chain* if there exists σ s.t. for all j:

$$t_j\sigma \to^*_{\mathcal{R}} s_{j+1}\sigma$$

Example - Dependency Chain

$\langle A(s(x_1), s(y_1)), A(x_1, a(s(x_1), y_1))\rangle, \langle A(s(x_2), s(y_2)), A(x_2, a(s(x_2), y_2))\rangle$

Consider $\sigma = \{x_1/s(0), y_1/0, x_2/0, y_2/a(s(0), 0)\}$, then

$(A(x_1, a(s(x_1), y_1)))\sigma \to^* (A(s(x_2), s(y_2)))\sigma$

Termination by Dependency Pairs

Termination Criterion by Dependency Pairs [T. Arts, J. Giesl 1997]

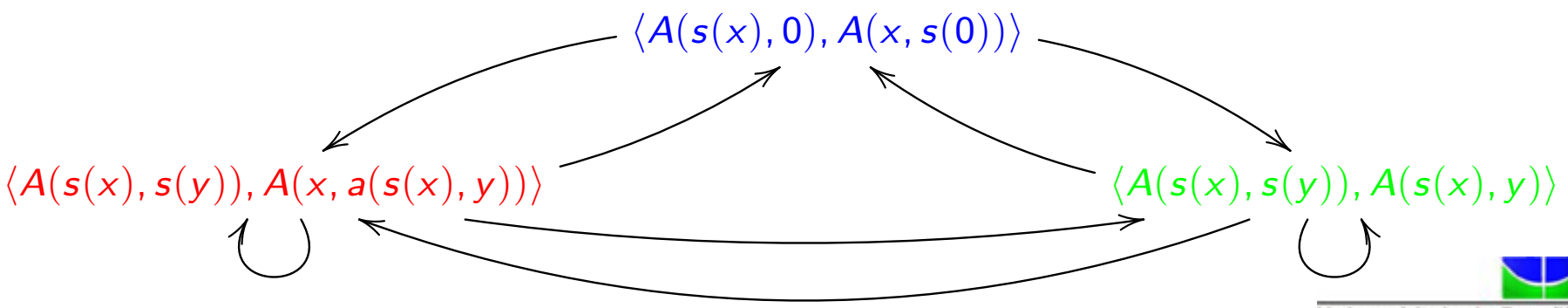
A TRS $\mathcal{R}$ is terminating if and only if there exists no infinite dependency chain.

Termination by Dependency Pairs

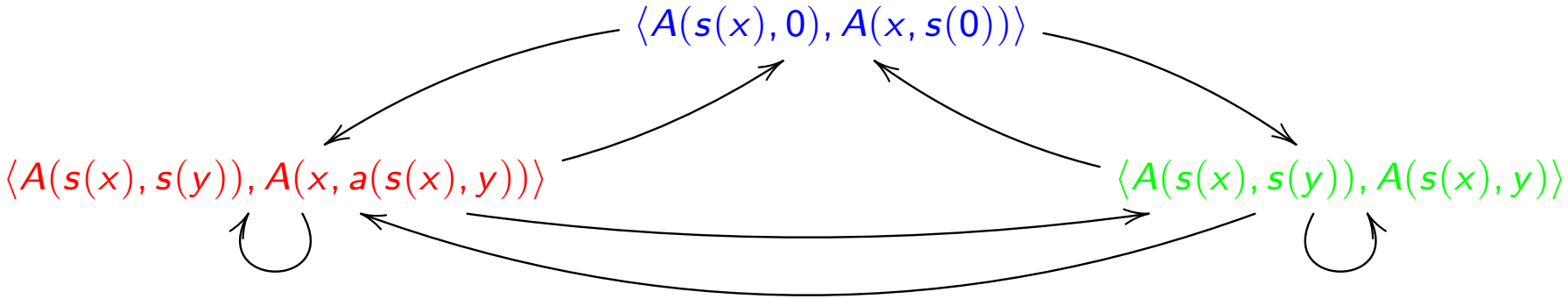
Estimated dependency graph [T. Arts, J. Giesl]

Given $\mathcal{R} = (C, D, R)$, its estimated dependency graph is the directed graph whose nodes are the DPs of $\mathcal{R}$ and there is and arc from $\langle s, t\rangle$ to $\langle v, w\rangle$ if REN(CAP(t)) and v are unifiable, where:
CAP(u) replaces each subterm rooted by a defined symbol in u by variables and
REN(u) replaces each variable in u by a fresh variable.

Estimated dependency graph for Ackermann



Termination by Dependency Pairs



REN(CAP($A(x, a(s(x), y))$)) = $A(x_1, y_1)$ that unifies with $A(s(x), 0)$;

REN(CAP($A(x, s(0))$)) = $A(x_2, s(0))$ that does not unify with $A(s(x), 0)$.

Termination by Dependency Pairs

Modular termination via DPs [T. Arts, J. Giesl]

A TRS $\mathcal{R}$ is terminating iff for each cycle $\mathcal{P}$ in the estimated depencency graph there exists no infinite dependency chain from $\mathcal{P}$.

Proof: if $\mathcal{R}$ non terminating, there exists an infinite dependency chain which should repeat infinitely one DP, say $\langle s, t\rangle$. The infinite chain has the form

$$\cdots \langle s\rho_1, t\rho_1\rangle \cdots \langle s\rho_2, t\rho_2\rangle \cdots \langle s\rho_3, t\rho_3\rangle \cdots$$

and its tail is and infinite dependency chain from one cycle in the graph. □

Termination by Dependency Pairs

Modular termination via DPs [T. Arts, J. Giesl]

A TRS $\mathcal{R} = (\mathcal{C}, \mathcal{D}, \mathcal{R})$ is terminating if for each cycle $\mathcal{P}$ in the estimated depencency graph there exists a well-founded weakly monotonic quasi-ordering $\geq_\mathcal{P}$ where both $\geq_\mathcal{P}$ and $>_\mathcal{P}$ are closed under substitution, such that

- $l \geq_\mathcal{P} r$ for all $l \to r \in R$;
- $s \geq_\mathcal{P} t$ for all DPs in $\mathcal{P}$, and
- $s >_\mathcal{P} t$ for at least one DP in $\mathcal{P}$.

Proof: Supposing the existence of an infinite dependency chain from a cycle $\mathcal{P}$ gives a contradiction on the well-foundedness of $>_\mathcal{P}$. □

Dependency Pairs for PVS0 functions

- The dependency pairs here are calling contexts;
- To relate CCs in chains, instead rewriting semantic evaluation should be applied.
- Two CCs are linked if it is possible to "reach" the second evaluating some input value that gives as output the second one.
- If only one function is defined by rewriting, CCGs can be then straightforwardly related with DPs for PVS0 functions.

Summary

- Straightforward formalization of equivalence between SCP and DP via CCG for PVS0 functional programs.
- (Innermost) semantic evaluation corresponds to normalization: A sequence of DPs $\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, \langle s_3, t_3\rangle \cdots$ is a *Dependency Chain* if there exists σ s.t. for all j:

$$t_j\sigma \to^{!}_{\mathcal{R}} s_{j+1}\sigma$$

Work in progress on DP

- Formalization of equivalence by partial evaluation:
  A sequence of DPs $\langle s_1, t_1\rangle, \langle s_2, t_2\rangle, \langle s_3, t_3\rangle \cdots$ is a *Dependency Chain* if there exists σ s.t. for all j:

$$t_j\sigma \to^*_{\mathcal{R}} s_{j+1}\sigma$$

- Formalization of theorems of modular termination via DPs.

References


Arts, T. and Giesl, J. (1998).
Modularity of Termination using Dependency Pairs.
In *RTA*, volume 1379 of *LNCS*, pages 226–240.


Arts, T. and Giesl, J. (2000).
Termination of Term Rewriting using Dependency Pairs.
Theoretical Computer Science 236(1-2):133–178.

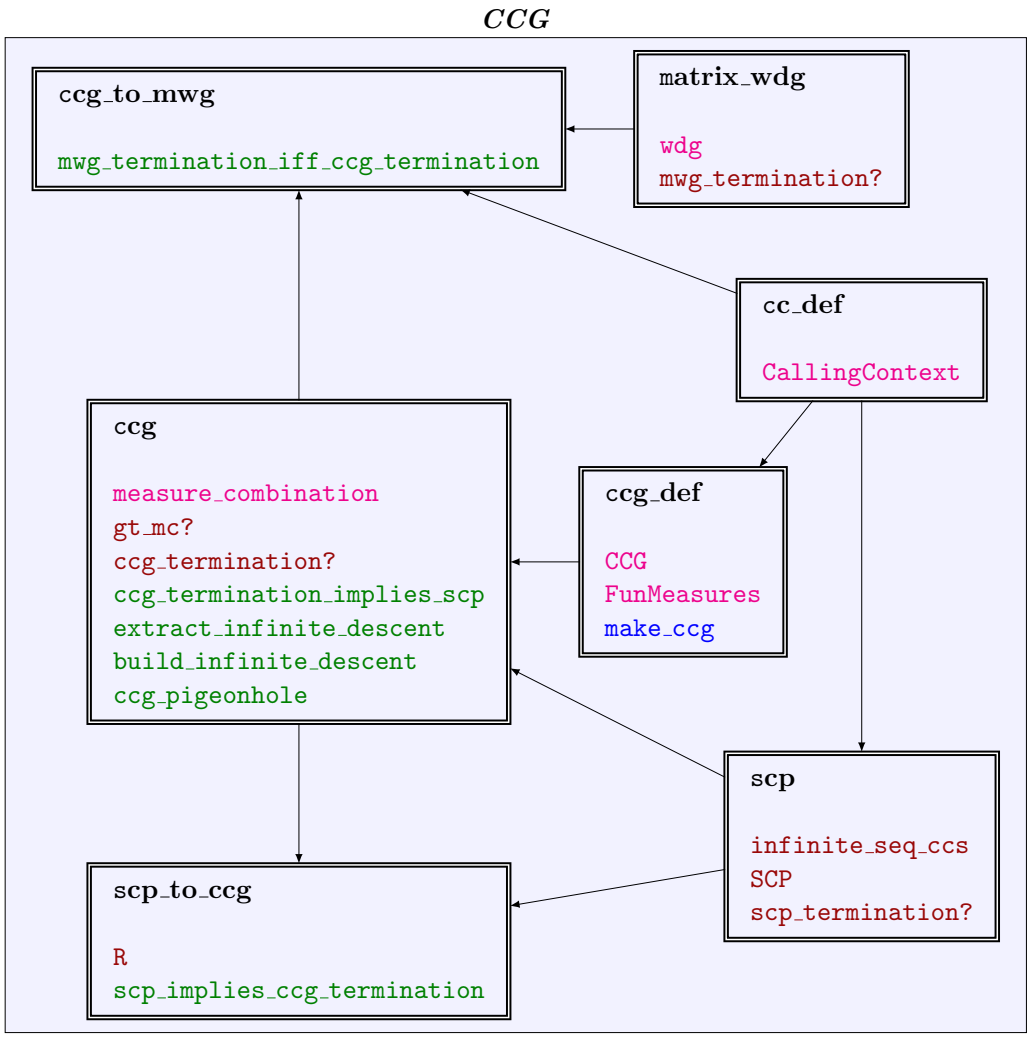

C.S. Lee, N.D. Jones, and A.M. Ben-Amram. (2001)
The size-change principle for program termination.
In *ACM* proc. *POPL*, pages 81–92.


P. Manolios and D. Vroon. (2006)
Termination analysis with calling context graphs.
In *CAV*, volume 4144 of *LNCS*, pages 401–414.

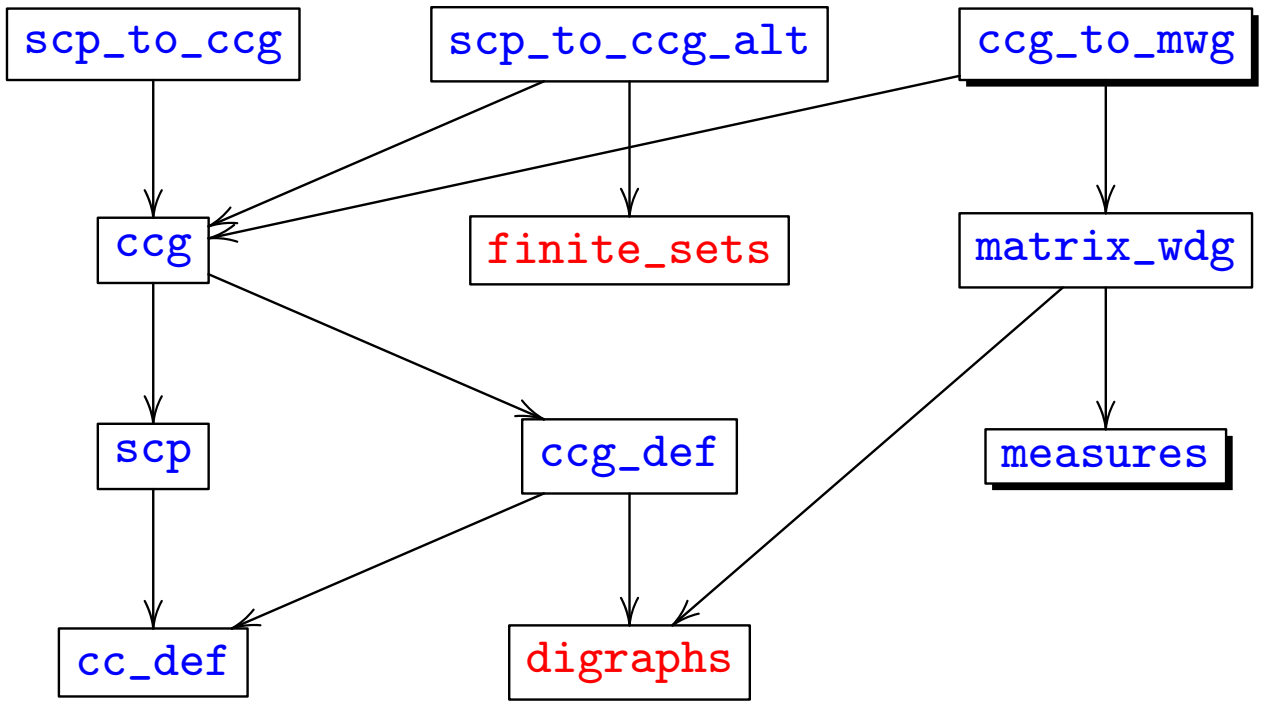

Giesl, J. and Thiemann, R. (2005).
The size-change principle and dependency pairs for termination of term rewriting.
Appl. Algebra Eng. Commun. Comput. 16(4): 229-270.


Iborra, J. and Nishida, N. and Vidal, G. and Yamada, A. (2017)
Relative Termination via Dependency Pairs.
Journal of Automated Reasoning 58(3): 391-411.

The Development: PVS theories CCG *and* PVSO



The PVS development - hierarchy of the CCG *theory*



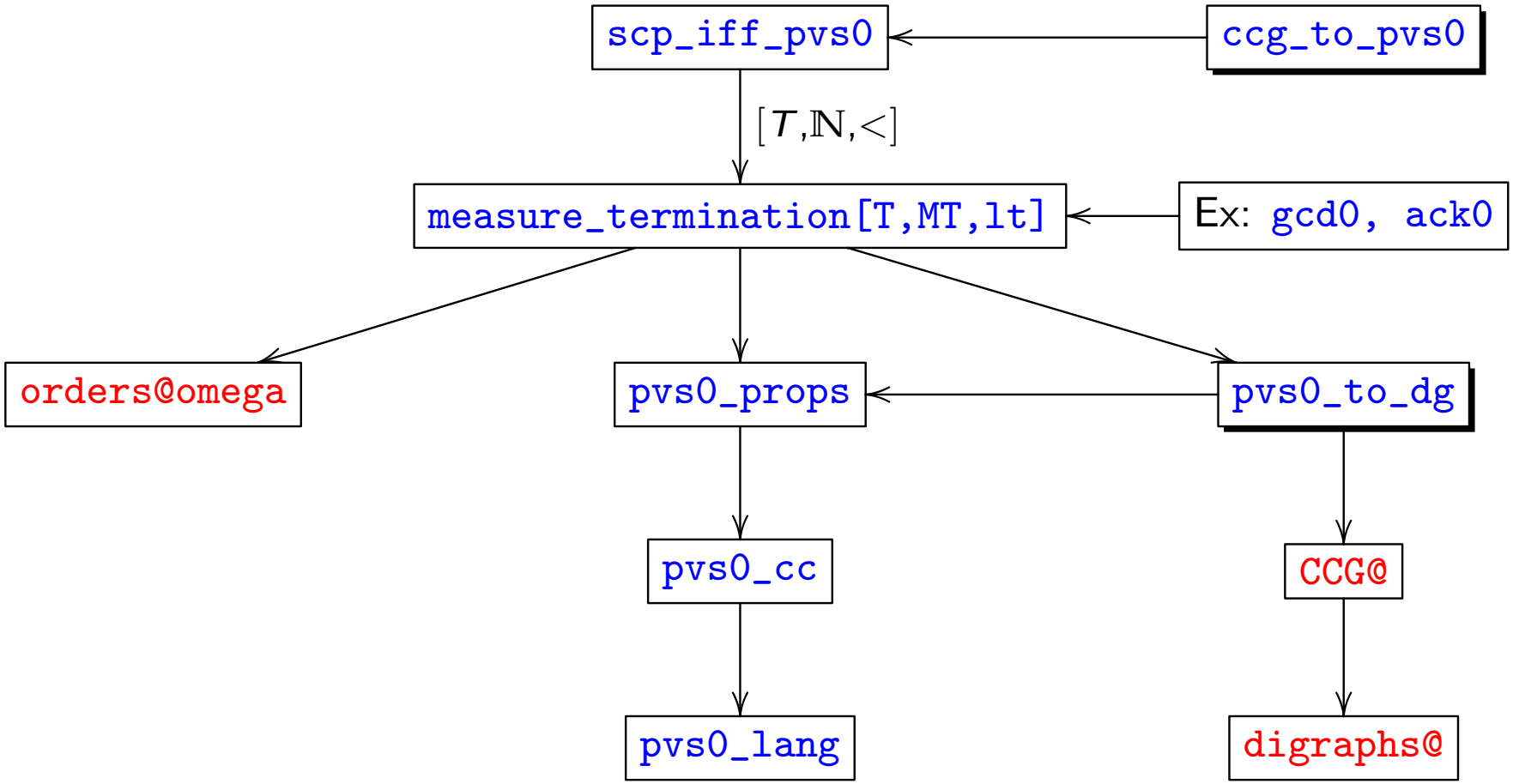
Available as part of the NASA LaRC PVS library (348 proofs):

http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html

PVS0


pvs0_expr
eval_expr
semantic_rel_expr
terminates_expr
eval_expr_termination
eval_expr_ge_n_j
eval_expr_semantic_rel_j
semantic_rel_eval_expr
eval_expr_semantic_rel
eval_expr_terminates
PVS0Expr
PVS0Expr
pvs0_undecidability
pvs0_undecidability
pvs0_comp
pvs0_sim
minimizar
composition
minimization
pvs0_lang
PVS0
determined?
mu
pvs0_cc
PVS0Expr_CC
valid_path
pvs0_tcc_valid_cc
eval_conds_rec
subterm_at
path_conditions
pvs0_termination
lt_val
omega_is_eval_ub
pvs0_tcc_implies_terminates
measure_termination
WFM
pvs0_tcc_termination
mu_soundness
terminates_implies_pvs0_tcc
pvs0_props
rec_mu_decreasing
pvs0_to_ccg
scp_implies_ccg_pvs0
pvs0_tcc_implies_ccg
scp_iff_pvs0
scp_termination_pvs0
pvs0_tcc_implies_scp
scp_implies_pvs0_tcc
pvs0_to_dg
sound_ccg_digraph
pvs0_to_fully_connected_dg
ccg_to_pvs0
ccg_termination_pvs0
ccg_implies_scp_pvs0
ccg_implies_pvs0_tcc
dp_termination
dep_link?
infinite_dp_chain?
dp_termination?
dp_decreasement?
lt_wfm_dp
dp_dec_implies_dp_termination
dg_termination
ltval_dg
leval_dg
sound_dpg_digrah
dg_termination?
dg_termination_implies_ccg
tcc_to_dg_termination
tcc_implies_dg_termination
ccg_implies_dg_termination
dp_to_tcc
dp_termination_implies_scp
dp_termination_implies_dp_dec

The PVS development - hierarchy of the PVS0 *theory*



Available as part of the NASA LaRC PVS library (366 proofs):

http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html