SizeChangeTool: A Termination Checker for Rewriting Dependent Types

Guillaume Genestier

Friday, June 28, 2019

Contents

Dedukti is a type-checker for the $\lambda\Pi$-calculus modulo rewriting.

Conversion:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash B : s \qquad A \leftrightarrow_{\beta\mathcal{R}} B}{\Gamma \vdash t : B}$$

Application:

$$\frac{\Gamma \vdash t : \Pi(x : A).B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B\,[x \mapsto u]}$$

```
(; Natural numbers ;)
Nat : Type.
0 : Nat.
s : Nat -> Nat.

(; Addition over natural numbers ;)
def plus : Nat -> Nat -> Nat.
[x]   plus x     0     --> x.
[x,y] plus (s x) y     --> s (plus x y).

(; Type with variable number of arguments ;)
def F : Nat -> Type.
[]  F 0     --> Nat.
[n] F (s n) --> Nat -> F n.

(; Variable-size summation ;)
def sum : (n: Nat) -> F n.
[]  sum 0          --> 0.
[]  sum (s 0)      --> x => x.
[n] sum (s (s n)) --> x => y => sum (s n) (plus x y).
```

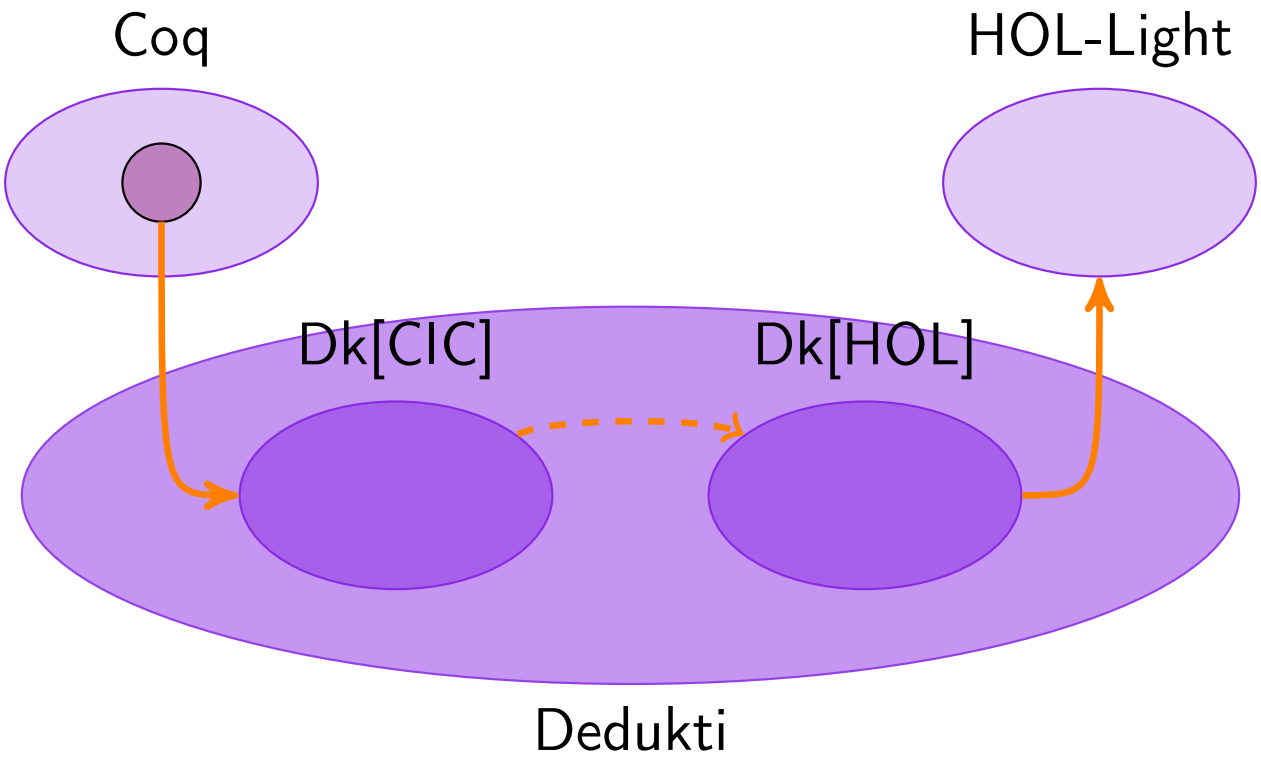
```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 3__VariableAritySummation.dk
YES
[Termin] 3__VariableAritySummation.dk was proved terminating using Dependency Pairs and SCT
```

```
(; Natural numbers ;)
Nat  : Type.
0 : Nat.
s : Nat -> Nat.

(; A few arithmetic functions ;)
def pred : Nat -> Nat.
[n] pred (s n) --> n.

def plus : Nat -> Nat -> Nat.
[n]    plus 0     n     --> n
[m, n] plus (s m) n     --> s (plus m n)
[n]    plus n     0     --> n
[m, n] plus m     (s n) --> s (plus m n).

[m,n,p] plus (plus m n) p --> plus m (plus n p).

def plus_rec_term : Nat -> Nat -> Nat.
[n]    plus_rec_term 0     n --> n
[m, n] plus_rec_term (s m) n --> plus_rec_term m (s n).
```

```
(; More functions ;)

def mult : Nat -> Nat -> Nat.
[]      mult 0      _ --> 0
[m, n] mult (s m) n --> plus n (mult m n).

def factorial : Nat -> Nat.
[]  factorial 0      --> s 0.
[n] factorial (s n) --> mult (s n) (factorial n).

def minus : Nat -> Nat -> Nat.
[x]   minus x      x      --> 0.
[]    minus 0      _      --> 0.
[x]   minus x      0      --> x.
[x,y] minus (s x) (s y) --> minus x y.

[x,y,z] minus (minus x y) z --> minus x (plus y z).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 4__Arith.dk
YES
[Termin] 4__Arith.dk was proved terminating using Dependency Pairs and SCT
```

Dedukti is well-suited for interoperability [Thiré18]



```
(; Type of the code of simple types ;)
typ : Type.

(; Decoding function ;)
T : typ -> Type.

(; Encoding of arrows ;)
arrow : typ -> typ -> typ.

(; Constructions of the lambda calculus ;)
lambda : (a : typ) -> (b : typ) -> (T a -> T b) -> T (arrow a b).
def appli : (a: typ) -> (b: typ) -> T (arrow a b) -> T a -> T b.

(; Beta rule ;)
[a,b,f,x] appli a b (lambda _ _ f) x --> f x.
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 5a_SimplyTypedLambda.dk
YES
[Termin] 5a_SimplyTypedLambda.dk was proved terminating using Dependency Pairs and SCT
```

```
(; No more codes ;)


(; Decoding function ;)
T : Type.

(; Encoding of arrows ar not required anymore ;)


(; Constructions of the lambda calculus ;)
lambda : (T -> T) -> T.
def appli  : T -> T -> T.

(; Beta rule ;)
[f,x] appli (lambda f) x --> f x.
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 5b_PureLambda.dk
MAYBE
[Warning] SizeChangeTool was unable to prove 5b_PureLambda.dk terminating
 * 5b_PureLambda.lambda is not positive in the rules
  - Rule noname.appli!15
```

Contents

We define $[\![.]\!]$ as the fixpoint of a monotonic function.

Lemma (Adequacy)

Define $[\![T]\!]$ *such that:*

- *If for all* f, $f \in [\![\Theta_f]\!]$ *and* $\Gamma \vdash t : T$, *then* $t \in [\![T]\!]$;
- $t \in [\![T]\!]$ *implies* $t \in \mathrm{SN}(\rightarrow_{\beta\mathcal{R}})$.

Definition (Dependency Pairs)

A rule $f\ \bar{l} \to r$ gives rise to the *dependency pairs* $f\ \bar{l} > g\ \bar{m}$ where:

- g is (partially) defined by rewriting,
- $g\ \bar{m}$ is a maximally applied subterm of r.

Higher-Order

Static and dynamic definition: [Blanqui06][Kusakari, Sakai 07][Kop, van Raamsdonk 12][Kop, Fuhs 19]

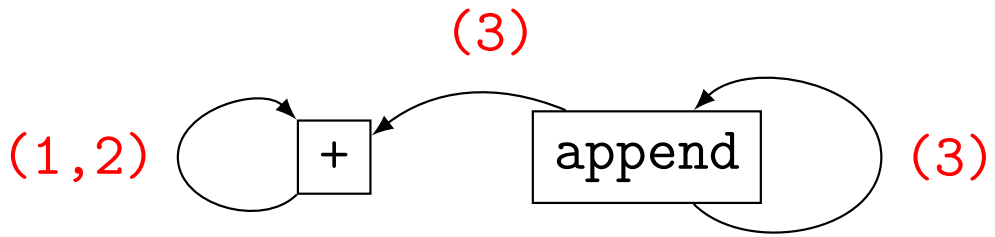
Example

```
def plus : Nat -> Nat -> Nat.
set infix "+" := plus.
[q]     0      + q      --> q.
[p,q]  (S p)   + q      --> S (p + q). (1)
[p,q]  p       + (S q)  --> S (p + q). (2)

def append: (p: Nat) -> List p ->
            (q: Nat) -> List q -> List (p + q).
[q,m]       append _ nil           q m --> m.
[x,p,l,q,m] append _ (cons x p l) q m -->
             cons x (p + q) (append p l q m). (3)
```

```
(1)           (S p) + q      > p + q
(2)           p      + (S q) > p + q
(3) append _ (cons x p l) q m > append p l q m
(3) append _ (cons x p l) q m > p + q
```

Call-Graph: Example

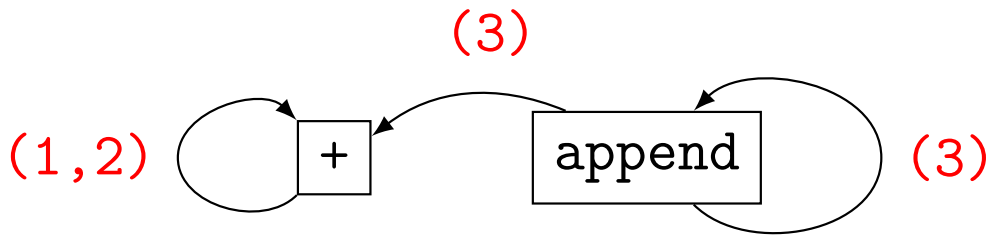
```
def plus : Nat -> Nat -> Nat.
set infix "+" := plus.
[q]     0     + q     --> q.
[p,q]  (S p)  + q     --> S (p + q). (1)
[p,q]  p      + (S q) --> S (p + q). (2)

def append: (p: Nat) -> List p ->
            (q: Nat) -> List q -> List (p + q).
[q,m]       append _ nil           q m --> m.
[x,p,l,q,m] append _ (cons x p l) q m -->
             cons x (p + q) (append p l q m). (3)
```



Size-Change Termination : Example

Introduced in [Lee, Jones, Ben Amram, 02] and used for MLTT in [Wahlstedt07] (also used in [Abel, Altenkirch 02]).
Keeping track of the evolution of the sizes of the arguments:

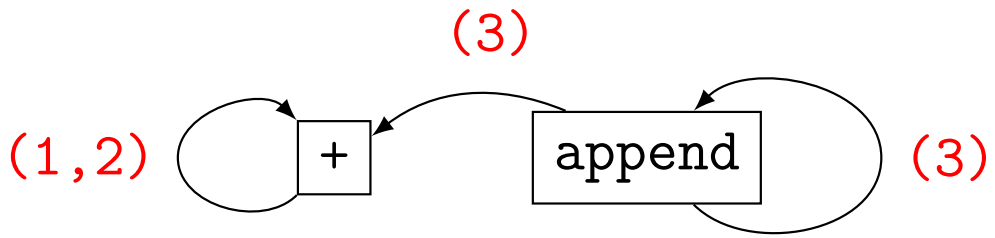
(1) `plus (S p) q > plus p q`	$\begin{array}{r} \\ \text{S p} \\ \text{q} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix} \end{array}$
(3) `append _ (cons x p l) q m > plus p q`	$\begin{array}{r} \\ \text{cons x p } \overline{\text{l}} \\ \text{q} \\ \text{m} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} \infty & \infty \\ < & \infty \\ \infty & = \\ \infty & \infty \end{pmatrix} \end{array}$



Size-Change Termination : Example

Introduced in [Lee, Jones, Ben Amram, 02] and used for MLTT in [Wahlstedt07] (also used in [Abel, Altenkirch 02]).
Keeping track of the evolution of the sizes of the arguments:

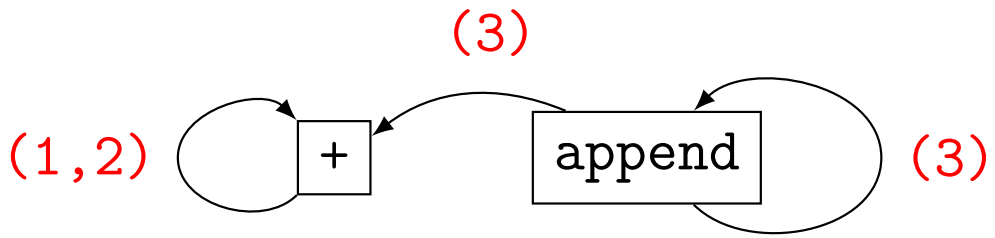
(1) `plus (S p) q > plus p q`	$\begin{array}{c} \\ \text{S p} \\ \text{q} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix} \end{array}$
(3) `append _ (cons x p l) q m > plus p q`	$\begin{array}{c} \\ \text{cons x p } \overline{\text{l}} \\ \text{q} \\ \text{m} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} \infty & \infty \\ < & \infty \\ \infty & = \\ \infty & \infty \end{pmatrix} \end{array}$



Size-Change Termination : Example

Introduced in [Lee, Jones, Ben Amram, 02] and used for MLTT in [Wahlstedt07] (also used in [Abel, Altenkirch 02]).
Keeping track of the evolution of the sizes of the arguments:

(1) `plus (S p) q > plus p q`	$\begin{array}{r} \\ \text{S p} \\ \text{q} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix} \end{array}$
(3) `append _ (cons x p l) q m > plus p q`	$\begin{array}{r} \\ \text{cons x p } \overline{\text{l}} \\ \text{q} \\ \text{m} \end{array} \begin{array}{c} \begin{array}{cc} \text{p} & \text{q} \end{array} \\ \begin{pmatrix} \infty & \infty \\ < & \infty \\ \infty & = \\ \infty & \infty \end{pmatrix} \end{array}$



```
(; Natural numbers ;)
Nat : Type.

0 : Nat.
s : Nat -> Nat.

(; The Ackermann function, not primitive recursive ;)
def Ackermann : Nat -> Nat -> Nat.
[n]    Ackermann 0     n     --> s n
[m]    Ackermann (s m) 0     --> Ackermann m (s 0)
[m, n] Ackermann (s m) (S n) --> Ackermann m (Ackermann (s m) n).

(; [Thiemann, Giesl 05] example, not handled by
classical path orderings ;)
def f : Nat -> Nat -> Nat.
[]    f 0     0     --> 0.
[x,y] f (s x) y     --> f x (s x).
[x,y] f x     (s y) --> f y x.
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 11a_Ackermann.dk
YES
[Termin] 11a_Ackermann.dk was proved terminating using Dependency Pairs and SCT
```

```
T : Type.

a : T.
b : T.

def f : T -> T -> T -> T.
[x] f a b x --> f x x x.

def coin : T -> T -> T.
[x,y] coin x y --> x.
[x,y] coin x y --> y.

(; This example is non-terminating:
f (coin a b) (coin a b) (coin a b) -->
f a          (coin a b) (coin a b) -->
f a          b          (coin a b) -->
f (coin a b) (coin a b) (coin a b)
;)
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 11b_Toyama.dk
MAYBE
[Warning] SizeChangeTool was unable to prove 11b_Toyama.dk terminating
 * 11b_Toyama.f is self looping in the rules
  - Rule noname.f!7
```

Main Result

Theorem

$\to_{\beta\mathcal{R}}$ terminates on every typable term in $\lambda\Pi/\mathcal{R}$ if:

- *$\to_{\beta\mathcal{R}}$ is locally confluent and type preserving,*
- *$\mathcal{R}$ is well-structured and Plain Function Passing,*
- *$\mathcal{R}$ is Size-Change Terminating.*

Well-Structuring

$\succeq$ quasi-order on the signature compatible with rewriting and typing.

Definition (Well-Structured System)

$\mathcal{R}$ is *well-structured* if for all rule $(\Delta, f\ \bar{l} \to r)$, with $\Theta_f = \Pi(\bar{x} : \bar{T}).U$, we have $\Delta \vdash_{\succeq f} r : U[\bar{x} \mapsto \bar{l}]$.

Definition (Plain Function Passing)

$f\ \bar{l} \to r$ is *PFP* if every functional type variable occurring in r is equal to one of the l_i.

```
def A : Type.
def T : A -> Type.

a : A.

[] A --> T a.
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 18a_NonStruct.dk
MAYBE
[Warning] SizeChangeTool was unable to prove 18a_NonStruct.dk terminating
[Termin] The file is not well-structured
```

```
Nat : Type.
Vec : Nat -> Type.
Elt : Type.

0 : Nat.
S : Nat -> Nat.

Nil : Vec 0.
Cons : (n : Nat) -> Elt -> Vec n -> Vec (S n).

def tail : (n : Nat) -> Vec (S n) -> Vec n.
[m,n,e,l] tail m (Cons n e l) --> tail m (Cons n e l).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ dkcheck 18b_NonRhsTyp.dk
[SUCCESS] File '18b_NonRhsTyp.dk' was successfully checked.
```

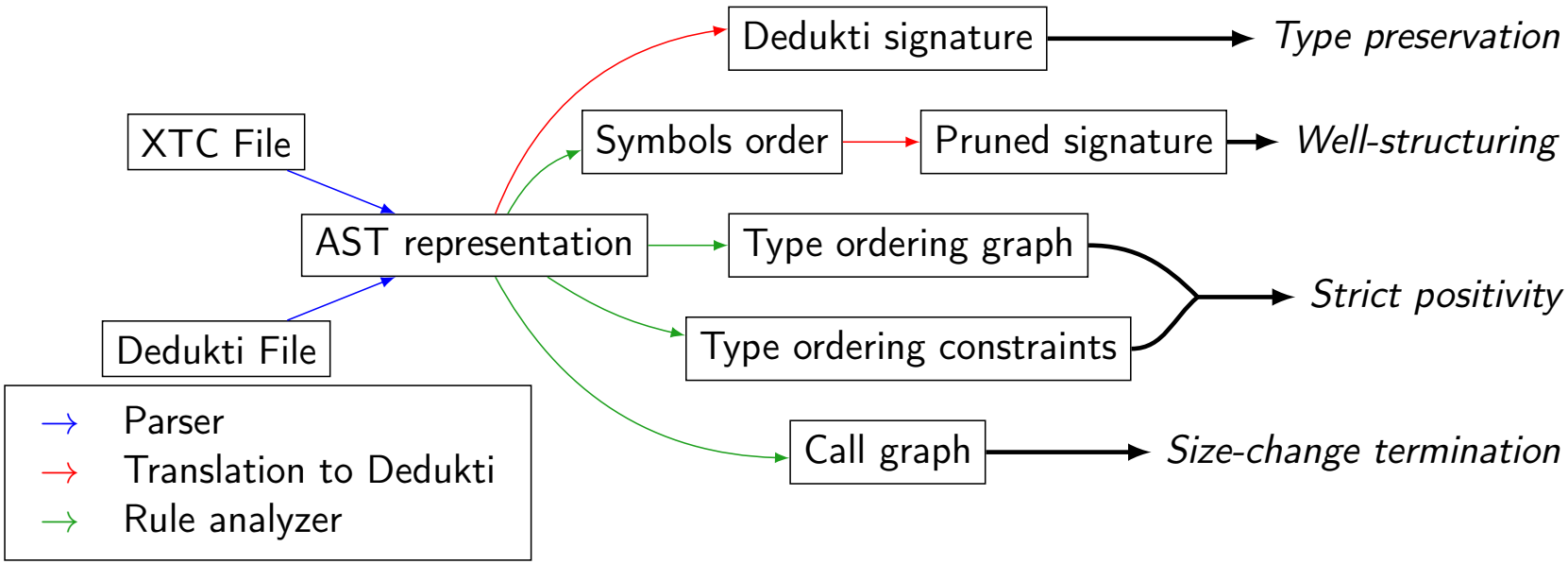
```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 18b_NonRhsTyp.dk
l[0] has type 18b_NonRhsTyp.Vec m[3], whether 18b_NonRhsTyp.Vec n[2] was inferred
MAYBE
[Warning] SizeChangeTool was unable to prove 18b_NonRhsTyp.dk terminating
 * 18b_NonRhsTyp.tail has untypable rhs in the rules
  - Rule noname.tail!12
 * 18b_NonRhsTyp.tail is self looping in the rules
  - Rule noname.tail!12
```

Contents

Theorem

$\rightarrow_{\beta\mathcal{R}}$ terminates on every typable term in $\lambda\Pi/\mathcal{R}$ if:

- *$\rightarrow_{\beta\mathcal{R}}$ is locally confluent and type preserving,*
- *$\mathcal{R}$ is well-structured and strictly positive,*
- *$\mathcal{R}$ is Size-Change Terminating.*



Type preservation

Checked by Dedukti assuming confluence.

Interaction with Dedukti

- A type-checker is required (*Well-Structuring*),
- Structures mainly shared,
- Still `dk_import.ml` and `dk_export.ml`, but very small.

Well-Structuring

The quasi-order

- Constructed using a graph transitive closure,
- Check of strict decrease in types.

$\Delta \vdash_{\preceq_f} r : U[\bar{x} \mapsto \bar{l}]$

- Pruned signature to check the right-hand side,
- Typability of the rhs is more than type preservation.

Strict positivity

- For real logic encodings,
- Require again to construct an order on type constructors,
- Plus subterm ordering (on normal form only),
- Not published yet, so `-pfp` option to be faithful with the paper.

```
Nat : Type.
Ord : Type.

0 : Nat.
s : Nat -> Nat.

0_ord : Ord.
s_ord : Ord -> Ord.
lim   : (Nat -> Ord) -> Ord.

A : Type.

def plus_ord : Ord -> Ord -> Ord.
[x]    plus_ord x 0_ord      --> x.
[x, y] plus_ord x (s_ord y) --> s_ord (plus_ord x y).
[x, f] plus_ord x (lim f)   --> lim (n => plus_ord x (f n)).

def ordrec : A -> (Ord -> A -> A) ->
             ((Nat -> Ord) -> (Nat -> A) -> A) -> Ord -> A.
[x, y, z]     ordrec x y z 0_ord      --> x.
[x, y, z, n] ordrec x y z (s_ord n) --> y n (ordrec x y z n).
[x, y, z, f] ordrec x y z (lim f)   -->
                                  z f (n => ordrec x y z (f n)).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 19a_BrouwerOrd.dk
YES
[Termin] 19a_BrouwerOrd.dk was proved terminating using Dependency Pairs and SCT
```

```
(; Encoding of Barendregt's Calculus of Construction
with 2 universes ;)

K : Type.
T : Type.

def eps : T -> Type.
def eta : K -> Type.

t : K.
[] eta t --> T.

Pitt : a : T -> (eps a -> T) -> T.
Pitk : a : T -> (eps a -> K) -> K.
Pikt : a : K -> (eta a -> T) -> T.
Pikk : a : K -> (eta a -> K) -> K.

[a, b] eps (Pitt a b) --> x : eps a -> eps (b x).
[a, b] eta (Pitk a b) --> x : eps a -> eta (b x).
[a, b] eps (Pikt a b) --> x : eta a -> eps (b x).
[a, b] eta (Pikk a b) --> x : eta a -> eta (b x).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 19b_Coc2.dk
MAYBE
[Warning] SizeChangeTool was unable to prove 19b_Coc2.dk terminating
 * 19b_Coc2.Pikk is not positive in the rules
  - Rule noname.eta!20
```

```
(; Encoding of Barendregt's Lambda-P-Omega PTS ;)

K : Type.
T : Type.

def eps : T -> Type.
def eta : K -> Type.

t : K.
[] eta t --> T.

Pitt : a : T -> (eps a -> T) -> T.
Pitk : a : T -> (eps a -> K) -> K.
(; Pikt : a : K -> (eta a -> T) -> T. ;)
Pikk : a : K -> (eta a -> K) -> K.

[a, b] eps (Pitt a b) --> x : eps a -> eps (b x).
[a, b] eta (Pitk a b) --> x : eps a -> eta (b x).
(; [a, b] eps (Pikt a b) --> x : eta a -> eps (b x). ;)
[a, b] eta (Pikk a b) --> x : eta a -> eta (b x).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 19c_LamPOmega.dk
YES
[Termin] 19c_LamPOmega.dk was proved terminating using Dependency Pairs and SCT
```

```
(; Encoding of the full predicative PTS with 2 universes ;)

K : Type.
T : Type.

def eps : T -> Type.
def eta : K -> Type.

t : K.
[] eta t --> T.

Pitt : a : T -> (eps a -> T) -> T.
Pitk : a : T -> (eps a -> K) -> K.
Pikt : a : K -> (eta a -> T) -> K.
Pikk : a : K -> (eta a -> K) -> K.

[a, b] eps (Pitt a b) --> x : eps a -> eps (b x).
[a, b] eta (Pitk a b) --> x : eps a -> eta (b x).
[a, b] eta (Pikt a b) --> x : eta a -> eps (b x).
[a, b] eta (Pikk a b) --> x : eta a -> eta (b x).
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native 19d_FullPredicativePTS.dk
YES
[Termin] 19d_FullPredicativePTS.dk was proved terminating using Dependency Pairs and SCT
```

```
ggenestier@latitude7480:~/hor2019/Examples$ ~/SizeChangeTool/sct.native --pfp 19d_FullPredicativePTS.dk
MAYBE
[Warning] SizeChangeTool was unable to prove 19d_FullPredicativePTS.dk terminating
 * 19d_FullPredicativePTS.eps is not PFP in the rules
   - Rule noname.eps!17
```

Size-Change Termination

- Mostly uses Lepigre's implementation for PML_2;
- 2 distinct steps: Extraction of the calls, transitive closure of the call-graph;
- A lot of administrative shifting of de Bruijn indices.

Size of the code

37	rules.ml
128	sign.ml
171	sizematrix.ml
89	dk_export.ml
75	dk_import.ml
38	pfp_checker.ml
325	positivity_checker.ml
98	rhs_typability.ml
107	call_extractor.ml
150	callgraph.ml
40	sizechange.ml
16	arity_checker.ml
231	sct.ml
281	tpdb_to_dk.ml
1786	total

Comparisons with Other Tools

Simply-typed

- Annual competition, few participants (Wanda, SOL?),
- Prove less examples, much faster.

Orthogonal Rules

- Integrated in proof assistants,
- Very similar to Agda's,
- Easily deals with argument permutation, unlike Coq's.

Future Work

Plain Function Passingness

Publish a proof corresponding to what is actually implemented.

Dependency Pairs

- Adapt more "processors",
- Recover completeness.

Tool improvement

- Modularity results:
  - with simple types (like [Harper, Honsell, Plotkin 93]),
  - with first-order (like [Jouannad, Okada97] and [Fuhs, Kop11]),
- Non-termination,
- Other input format (Agda).

Downloading

SizeChangeTool and Dedukti are open-source:

https://github.com/Deducteam/SizeChangeTool