Proving termination using the Size-Change Principle

Guillaume Genestier

Olivier Hermant (MINES ParisTech) and Frédéric Blanqui (Inria)

Friday, September 15th 2017

Dedukti is a multipurpose type-checker based on the $\lambda\Pi$-calculus modulo theory.

Example of rewrite rule

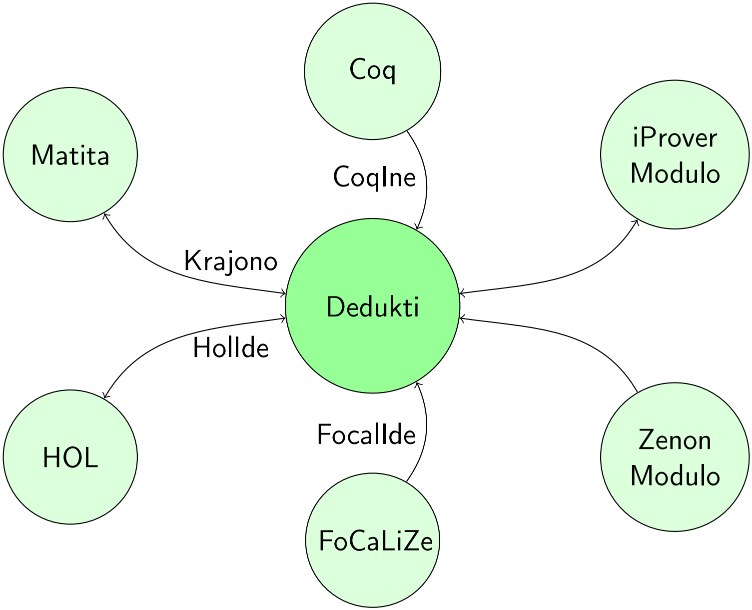
```
Nat : Type.
0 : Nat.
S : Nat -> Nat.

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

Example of dependent type

```
List : Nat -> Type.
nil : List 0.
Cons : (n:Nat) -> Nat -> List n -> List (S n)
```

Dedukti is well-suited for interoperability



We consider a set of rewrite rules.

```
Nat : Type.

0 : Nat.
S : Nat -> Nat.

def plus : Nat -> Nat -> Nat.
[n] plus 0 n --> n

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

def mult : Nat -> Nat -> Nat.
[] mult 0 _ --> 0

[m,n] mult (S m) n --> plus n (mult m n).
```

Size-Change Principle [Lee, Jones, Ben-Amram, 2001] [Lepigre, Raffalli, 2017]

We consider a set of rewrite rules.

`Nat : Type.`	
`0 : Nat.` `S : Nat -> Nat.`	

`def plus : Nat -> Nat -> Nat.`
`[n] plus 0 n --> n`
`[m,n] plus (S m) n --> S (plus m n).`
	$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$

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

Size-Change Principle [Lee, Jones, Ben-Amram, 2001] [Lepigre, Raffalli, 2017]

We consider a set of rewrite rules.

`Nat : Type.`	
`0 : Nat.` `S : Nat -> Nat.`	

`def plus : Nat -> Nat -> Nat.`
`[n] plus 0 n --> n`
`[m,n] plus (S m) n --> S (plus m n).`
	$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$

`def mult : Nat -> Nat -> Nat.`
`[] mult 0 _ --> 0`
`[m,n] mult (S m) n --> plus n (mult m n).`
	$\begin{pmatrix} \infty & \infty \\ = & \infty \end{pmatrix}$

Size-Change Principle [Lee, Jones, Ben-Amram, 2001] [Lepigre, Raffalli, 2017]

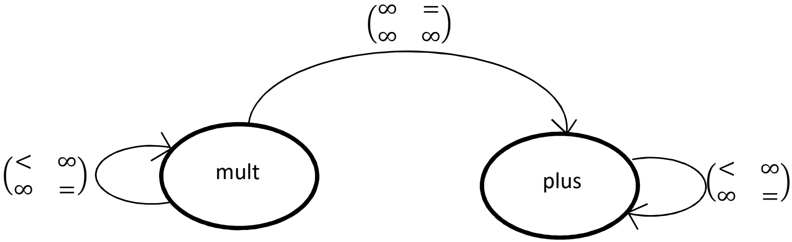
We consider a set of rewrite rules.

`Nat : Type.`	
`0 : Nat.` `S : Nat -> Nat.`	

`def plus : Nat -> Nat -> Nat.`
`[n] plus 0 n --> n`
`[m,n] plus (S m) n --> S (plus m n).`
	$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$

`def mult : Nat -> Nat -> Nat.`
`[] mult 0 _ --> 0`
`[m,n] mult (S m) n --> plus n (mult m n).`
	$\begin{pmatrix} \infty & \infty \\ = & \infty \end{pmatrix}$
	$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$

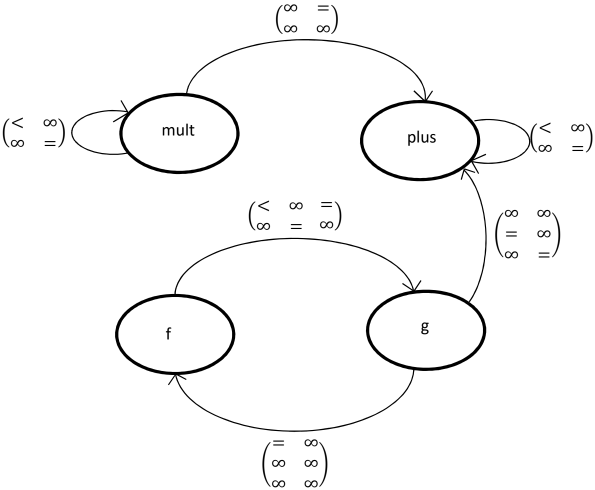
Call graph



Mutually recursive definition

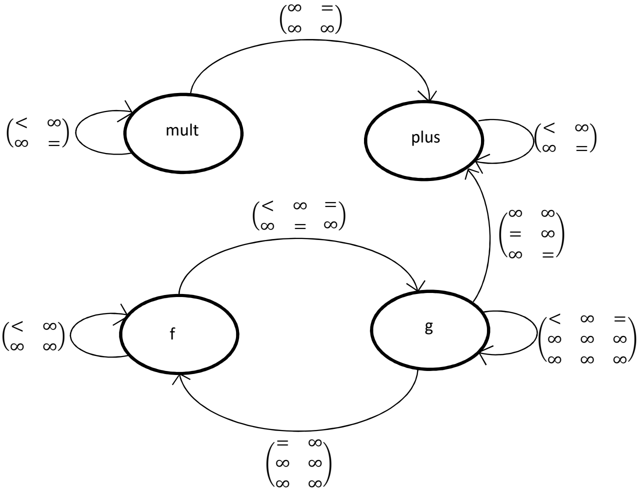
`def f : Nat -> Nat -> Nat.` `def g : Nat -> Nat -> Nat -> Nat.`	
`[x] f 0 x --> x` `[i,x] f (S i) x --> g i x (S i)`	$\begin{pmatrix} < & \infty & = \\ \infty & = & \infty \end{pmatrix}$
`[a,b,c] g a b c --> f a (plus b c)`	$\begin{pmatrix} = & \infty \\ \infty & \infty \\ \infty & \infty \end{pmatrix}$ $\begin{pmatrix} \infty & \infty \\ = & \infty \\ \infty & = \end{pmatrix}$

Call graph



mult
plus
f
g
$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$
$\begin{pmatrix} \infty & = \\ \infty & \infty \end{pmatrix}$
$\begin{pmatrix} < & \infty \\ \infty & = \end{pmatrix}$
$\begin{pmatrix} < & \infty & = \\ \infty & = & \infty \end{pmatrix}$
$\begin{pmatrix} \infty & \infty \\ = & \infty \\ \infty & = \end{pmatrix}$
$\begin{pmatrix} = & \infty \\ \infty & \infty \\ \infty & \infty \end{pmatrix}$

Transitive closure of the call graph



An implementation of this algorithm has been done for *Dedukti*.

`List : Type.`	
`Nil : List.` `Cons : Nat -> List -> List.`	
`def map : (Nat -> Nat) -> List -> List.`	

`[] map _ Nil --> Nil.`
`[f, x, l] map f (Cons x l) -->`
`Cons (f x) (map f l).`
	$\begin{pmatrix} = & \infty \\ \infty & < \end{pmatrix}$
	$\begin{pmatrix} \infty \\ = \end{pmatrix}$

`Lamt : Type.`	
`def App : Lamt -> Lamt -> Lamt.` `Lam : ( Lamt -> Lamt ) -> Lamt.`	
`[f,t] App (Lam f) t --> f t.`	$\begin{pmatrix} \infty \\ = \end{pmatrix}$

Structure of Wahlstedt thesis

- Define a reducibility predicate for weak normalisation.
- Show that if every function in the signature is reducible then every typable term is reducible.
- Show that if the call relation is well-founded and every type in the signature is reducible, then every function in the signature is reducible.
- The size-change principle is used to show that the call relation is well-founded.
- The reducibility of every type occuring in the signature is decidable.

Syntax

Definition (Terms)

We use :

- x,y,z *to denote variables,*
- f, g *to denote defined constants,*
- c *to denote element constructors,*
- d *to denote set constructors.*

$$t, \tau, u, v, l, r ::= x \mid \lambda(x : u).t \mid t\ u \mid \Pi(x : t)\ u \mid \text{Kind} \mid \text{Type} \mid c \mid d \mid f$$

Definition (Contexts)

$$\Gamma, \Delta ::= [] \mid \Gamma, x : t$$

Normalisation predicates

$$
\begin{aligned}
\mathsf{NF}(u) &\equiv \neg\left(\exists v. u \rightsquigarrow v\right)\\
\mathsf{SN}(u) &\equiv \neg\left(\exists (v_i)_{i \in \mathbb{N}}. v_0 = u \wedge \forall i. v_i \rightsquigarrow v_{i+1}\right)\\
\mathsf{WN}(u) &\equiv \exists v. u \rightsquigarrow^* v \wedge \mathsf{NF}(v)\\
u \Downarrow v &\equiv u \rightsquigarrow^* v \wedge \mathsf{NF}(v)
\end{aligned}
$$

Sub-categories of terms

Definition (β-normal term)

We define this syntactical sub-category of terms as :

$$s ::= x\, s_1 \dots s_n \mid h\, s_1 \dots s_{\mathrm{ar}(h)} \mid \lambda(x : T).s$$

where h is one symbol in the signature, set constructor, element constructor or defined function.

Definition (Constructor patterns)

$$p ::= x \mid c\, p_1 \dots p_n$$

Definition (Strongly neutral terms)

$$\begin{aligned} b ::= {} & x\, t_1 \dots t_n \textit{ where } \mathrm{NF}(t_i) \\ & \mid f\ t_1 \dots t_n \textit{ where } \mathrm{NF}(f\ t_1 \dots t_n) \textit{ and } n \geqslant \mathrm{ar}(f) \end{aligned}$$

Rewrite rules

Each *rewrite rule* is of the form $f\ p_1 \dots p_k \to s$ where :

- the p_i are constructor patterns,
- $k \leqslant \mathrm{ar}(f)$,
- p_k is not a variable,
- s is β-normal,
- s starts with $(\mathrm{ar}(f) - k)$ λ-abstractions,
- the rule is *left-linear*, meaning that a free variable cannot appear twice in $f\ p_1 \dots p_k$.

Furthermore, the set of rewrite rules is *non-unifiable*, meaning that for any two rules $l_1 \to r_1$ and $l_2 \to r_2$, there are no substitutions σ_1 and σ_2 such that $\sigma_1(l_1) = \sigma_2(l_2)$.

Property (Confluence)

Such a rewrite system is orthogonal, hence it is confluent.

The reducibility predicate

$\mathrm{RED}_{(\mathrm{Kind})}(t)$ holds if one of the following conditions occurs:

- $t = \mathrm{Type}$ and then $\mathrm{RED}_{(\mathrm{Type})}(u)$ holds if one of the following conditions occurs:
  - $\exists d, u_1, \ldots, u_m. \begin{cases} u \Downarrow d\; u_1 \ldots u_m \\ \mathcal{D}(d) = \Pi(x_1 : T_1)\ldots(x_k : T_k)\ \mathrm{Type} \quad \text{and} \\ \forall i.\, \mathrm{RED}_{(\mathrm{Type})}(T_i) \wedge \mathrm{RED}_{(T_i)}(u_i) \end{cases}$

    then $\mathrm{RED}_{(u)}(v)$ holds if one of the following conditions occurs:

    - $\exists c, v_1, \ldots, v_n. \begin{cases} v \Downarrow c\; v_1 \ldots v_n \\ \mathcal{C}(c) = \Pi(x_1 : U_1)\ldots(x_m : U_m)\,(d\; \tau_1 \ldots \tau_k) \\ \forall i.\, \mathrm{RED}_{(\mathrm{Type})}\left(U_i\left[{}^{v_1}/_{x_1}, \ldots, {}^{v_{i-1}}/_{x_{i-1}}\right]\right) \\ \forall i.\, \mathrm{RED}_{\left(U_i\left[{}^{v_1}/_{x_1}, \ldots, {}^{v_{i-1}}/_{x_{i-1}}\right]\right)}(v_i) \end{cases}$
    - $\exists b. v \Downarrow b$

The reducibility predicate

- $\exists A, B. \begin{cases} u \Downarrow \Pi(x : A)\ B \\ \text{RED}_{(\text{Type})}(A) \\ \forall a.\, \text{RED}_{(A)}(a) \Rightarrow \text{RED}_{(\text{Type})}\left(B\left[{}^{a}/_{x}\right]\right) \end{cases}$ and then
  $\text{RED}_{(u)}(v)$ holds if $\forall a.\, \text{RED}_{(A)}(a) \Rightarrow \text{RED}_{(B[{}^{a}/_{x}])}(v\ a)$
- $\exists b.u \Downarrow b$ and then $\text{RED}_{(u)}(v)$ holds if $\exists b'.v \Downarrow b'$

- $\exists A, B. \begin{cases} t = \Pi(x : A)\ B \\ \text{RED}_{(\text{Type})}(A) \\ \forall a.\, \text{RED}_{(A)}(a) \Rightarrow \text{RED}_{(\text{Kind})}\left(B\left[{}^{a}/_{x}\right]\right) \end{cases}$ and then
  $\text{RED}_{(\Pi(x:A)\,B)}(u)$ holds if $\forall a.\, \text{RED}_{(A)}(a) \Rightarrow \text{RED}_{(B[{}^{a}/_{x}])}(u\ a)$

Typability, reducibility and weak normalisation

Proposition

For all T, t, if $\mathrm{RED}_{(\mathrm{Type})}(T)$ *then*

1. $\mathrm{RED}_{(T)}(t) \Rightarrow \mathrm{WN}(t)$
2. $t \Downarrow b \Rightarrow \mathrm{RED}_{(T)}(t)$

Theorem

If $\forall f.\, \mathrm{RED}_{(\mathrm{Type})}(\mathcal{F}(f)) \wedge \mathrm{RED}_{(\mathcal{F}(f))}(f)$ *then*

$$\Gamma \vdash t : T \Rightarrow \left[\forall \sigma.\, \mathrm{RED}_{\Gamma(x)}(\sigma(x)) \Rightarrow \mathrm{RED}_{(\sigma(T))}(\sigma(t))\right]$$

Call relation

Definition (Formal call)

We define $(f,(p_1,\dots,p_m)) \succ (g,(u_1,\dots,u_n))$ *by :*

- *there is a* k *such that* $f\ p_1 \dots p_k \to s$ *is in* $\mathbb{R}$,
- $\mathrm{ar}(f) = m$, $\mathrm{ar}(g) = n$,
- $g\ u_1 \dots u_n$ *is a subterm of* $s\ p_{k+1} \dots p_m$.

Definition (Instantiated call)

$(f,(t_1,\dots,t_m)) \tilde{\succ} (g,(v_1,\dots,v_n))$ *holds if there exists a substitution* σ *such that :*

- $\forall i. \exists p_i. t_i \leadsto^* \sigma(p_i)$,
- $\forall i. \mathrm{WN}(t_i)$,
- $\forall j. v_j = \sigma(u_j)$
- $(f,(p_1,\dots,p_m)) \succ (g,(u_1,\dots,u_n))$.

Final result

Theorem

If $\widetilde{\succ}$ *is well-founded and* $\forall f.\, \mathrm{RED}_{(\mathrm{Type})}(\mathcal{F}(f))$ *then*

$$\forall f.\, \mathrm{RED}_{(\mathcal{F}(f))}\,(f)$$

Structure of Wahlstedt thesis

- Define a reducibility predicate for weak normalisation.
- Show that if every function in the signature is reducible then every typable term is reducible.
- Show that if the call relation is well-founded and every type in the signature is reducible, then every function in the signature is reducible.
- The size-change principle is used to show that the call relation is well-founded.
- The reducibility of every type occurring in the signature is decidable.

Future work

- Strong normalisation (from Frédéric Blanqui's work)
- Study decidability of type reducibility
- Enrichment of the Size-Change Principle
- Implementation of the Wahlstedt criterion