

The syntactic guard condition of Coq

Bruno Barras

February 2, 2010

Overview

- 1 Theory
 - Basic criterion
 - Extensions
- 2 Algorithm
 - Efficiency
- 3 Discussion
- 4 Attic

A short history of the syntactic guard criterion

Prehistory:

- Recursion was based on [recursors](#) (Gödel's T, impredicative encodings).
- Only allows recursive calls on [direct subterms](#)
- Awkward in a functional programming setting!

Example

```
Definition half n :=
  fst(Rec (0,false)
        (fun (k,odd) => if odd then (k+1,false)
                          else (k,true))
        n)
```

instead of

```
Fixpoint half n :=
  match n with S(S k) => half k | _ => 0 end
```

A short history of the syntactic guard criterion

History:

- Proposal by Coquand (92):

`recursor = pattern-matching + fixpoint`

- Gimenez' paper (94): still the [official reference](#)

Translation towards recursors:

For $f : I \rightarrow T$, define I_f similar to I such that every subterm of type I comes with its image by f . Then write $g : I \rightarrow I_f$ and $h : I_f \rightarrow T$.

- Blanqui (05), Calculus of Algebraic Constructions: reducibility proof (CC + higher order rewriting)
- Only works for the basic criterion.

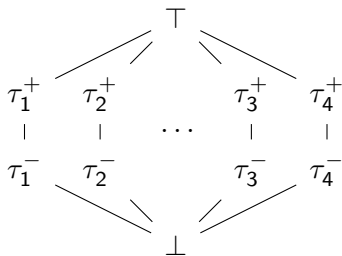
Basic criterion: size information

(non-strict) $\Sigma^+ ::= \top \mid \tau^+$ (\top = not a subterm)

(strict) $\Sigma^- ::= \perp \mid \tau^-$ (\perp = inaccessible cases)

(size info) $\Sigma ::= \Sigma^+ \cup \Sigma^-$

A map ρ associates size information to every variable



Regular trees as sets of paths

Positivity check:

- Checking there is no negative occurrence
- Identifies recursive positions: regular tree $\text{Str}(I, \vec{C})$
(not any constructor argument because of impredicativity)

Lemma

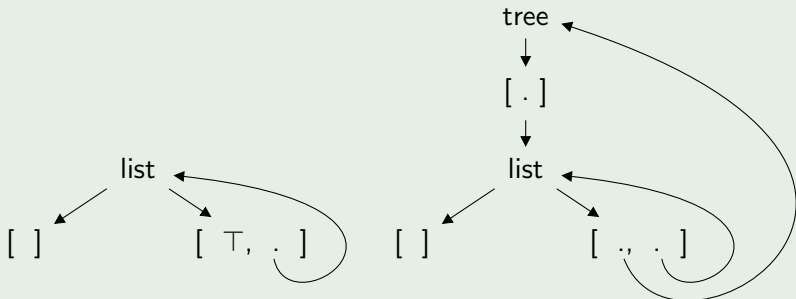
The computed tree is the set of paths that cannot contain an infinite number of inductive objects.

Regular trees: examples

- Different instances of the same inductive type may have different sets of recursive positions

Example ($\text{Str}(\text{list})$ and $\text{Str}(\text{tree})$)

Inductive tree:Set := Node(l:list tree).



Guard condition in short

- A judgement $\rho \vdash^S M \Rightarrow \sigma$ meaning that M has size information σ , where ρ associates size information to variables
- A judgement $M \in \text{Norm}_\rho^{F,k}$ meaning that M does recursive calls to F only on strict subterms, as specified by ρ
- Pattern-matching propagates information on pattern variables

$$\text{Constr}(i, l) \ x_1 \dots x_k \mid \sigma = \{(x_j, \sigma.i.j^-) \mid j \leq k\}$$

Remarks

- Easy encoding of recursors as fix+match (non regression)
- Bonus: allow recursive calls on deep subterms

Definition of the condition (1)

Definition (Terms)

$s \mid x \mid \Pi x : T. U \mid \lambda x : T. M \mid M N$
 $\mid \text{Ind}(X : A)\{\vec{C}\} \mid \text{Constr}(n, l) \mid \text{Fix } F_k : T := M$
 $\mid \text{Match } M \text{ with } \vec{p} \Rightarrow \vec{t} \text{ end}$

Typing rule:

$$\frac{\Gamma (F : T) \vdash M : T \quad M \in \text{Guard}_k^F}{\Gamma \vdash (\text{Fix } F_k : T := M) : T}$$

Initializing the termination check (Str() precomputed):

$$\frac{t_k = \text{Ind}(X : A)\{\vec{C}\} \vec{u} \quad \text{Str}(X, \vec{C}) = \tau \quad M \in \text{Norm}_{\{(x_k, \tau^+)\}}^{F,k}}{\lambda \vec{x} : \vec{t}. M \in \text{Guard}_k^F}$$

Definition of the condition (2)

$$\frac{M \in \text{Norm}_\rho^{f,k} \quad \rho \vdash^S M \Rightarrow \sigma \quad \forall i. b_i \in \text{Norm}_{\rho \cup \{p_i | \sigma\}}^{f,k}}{\text{Match } M \text{ with } \vec{p} \Rightarrow \vec{b} \text{ end} \in \text{Norm}_\rho^{f,k}}$$

$$\frac{\rho \vdash^S t_k \Rightarrow \sigma \quad \sigma \in \Sigma^- \quad \forall i, t_i \in \text{Norm}_\rho^{f,k}}{f \vec{t} \in \text{Norm}_\rho^{f,k}}$$

+ congruence rules...

Subterms

$$\frac{(x, \sigma) \in \rho}{\rho \vdash^S x \vec{t} \Rightarrow \sigma} \quad \frac{\rho \vdash^S M \Rightarrow \sigma}{\rho \vdash^S \lambda x : A. M \Rightarrow \sigma}$$

Checking guard modulo reduction

- Enable reusability of standard definitions

In fact, the typing rule for fixpoints is:

$$\frac{\Gamma (F : T) \vdash M : T \quad M \rightarrow_{\beta}^* M' \quad M' \in \text{Guard}_k^F}{\Gamma \vdash (\text{Fix } F_k : T := M) : T}$$

Breaks strong normalization!

Example

```
Fixpoint F n := let x := F n in 0.
```

```
Eval compute in (F 0).
```

Pattern-matching

- A match where all branches return a subterm is a subterm
- Enables inaccessible cases (absurd) elimination

$$\frac{\forall i, \rho \vdash^S b_i \Rightarrow \sigma_i}{\rho \vdash^S \text{Match } M \text{ with } \vec{p} \Rightarrow \vec{b} \text{ end} \Rightarrow \prod \vec{\sigma}}$$

Example

```
Definition pred n (H:n<>0) :=
```

```
  match n with
```

```
    0 => match H _ with end
```

```
  | S k => k
```

```
  end.
```

```
Fixpoint F x :=
```

```
  if eq_nat_dec x 0 then 0 else F (pred x)
```

Fixpoints as argument of F

- A fix returns a strict subterm if its body does
- Size information of recursive argument is propagated

$$\frac{\rho \vdash^S u_n \Rightarrow \sigma \quad \rho \cup \{(G, \tau^-), (x_n, \sigma)\} \vdash^S M \Rightarrow \tau^-}{\rho \vdash^S (\text{Fix } G_n : T := \lambda \vec{x} : \vec{t}. M) \vec{u} \Rightarrow \tau^-}$$

Example

```
Fixpoint F x y :=
  if "x ≤ y" then x else F (x-S(y)) y
```

Nested fixpoints

$$\frac{\rho \vdash^S u_n \Rightarrow \sigma \quad M \in \text{Norm}_{\rho\{(x_k, \sigma)\}}^{F,k} \quad T \in \text{Norm}_{\rho}^{F,k} \quad \vec{u} \in \text{Norm}_{\rho}^{F,k}}{(\text{Fix } G_n : T := M) \vec{u} \in \text{Norm}_{\rho}^{F,k}}$$

Example (size of a tree)

```
Fixpoint size (t:tree) :=
  match t with
  | Node l => fold_right (fun t' n => n+size t') 1 1
  end.
```

Overview

- 1 Theory
 - Basic criterion
 - Extensions
- 2 Algorithm
 - Efficiency
- 3 Discussion
- 4 Attic

Implementation

Follows the (almost) syntax-directed rules presented here

- `subterm_specif` implements $\rho \vdash^S M \Rightarrow \sigma$
- `check_one_fix` implements $M \in \text{Norm}_{\rho}^{f,k}$
 - succeed
 - fail
 - raise exception if partial application

Guard modulo reduction

- body is (δ -) reduced **on demand**: backtrack

Issue:

- Backtrack on constant expansion can be exponential
Fixpoint $F\ x := \text{id} (\text{id} (\text{id}\ F)) (\text{pred}\ x)$.

Propagation of size through match

- size of pattern variable is computed **eagerly**

$$\frac{M \in \text{Norm}_{\rho}^{f,k} \quad \rho \vdash^S M \Rightarrow \sigma \quad \forall i. b_i \in \text{Norm}_{\rho \cup \{p_i | \sigma\}}^{f,k}}{\text{Match } M \text{ with } \vec{p} \Rightarrow \vec{b} \text{ end} \in \text{Norm}_{\rho}^{f,k}}$$

Issue:

- Need to reduce terms to get the most precise information on match subject, but it is not used!

Fixpoint $F x := \text{if } \langle \text{long comp} \rangle \text{ then } F (\text{pred } x)$

Overview

- 1 Theory
 - Basic criterion
 - Extensions
- 2 Algorithm
 - Efficiency
- 3 Discussion
- 4 Attic

Syntactic criterion are fragile

- This condition has been extended over the years to support more schemes

Syntactic criterion are fragile

- This condition has been extended over the years to support more schemes
- Extension candidates:
 - “a term containing no subterm variable cannot be a subterm”
 - “no need to propagate size info for non-recursive types”

Syntactic criterion are fragile

- This condition has been extended over the years to support more schemes
- Extension candidates:
 - “a term containing no subterm variable cannot be a subterm”
 - “no need to propagate size info for non-recursive types”

Both are incomplete!

Syntactic criterion are fragile

- This condition has been extended over the years to support more schemes
- Extension candidates:
 - “a term containing no subterm variable cannot be a subterm”
 - “no need to propagate size info for non-recursive types”Both are incomplete!
- Bugs (or scary error messages)

Syntactic criterion are fragile

- This condition has been extended over the years to support more schemes
- Extension candidates:
 - “a term containing no subterm variable cannot be a subterm”
 - “no need to propagate size info for non-recursive types”

Both are incomplete!

- Bugs (or scary error messages)

Uncaught exception: `Assert_failure("kernel/inductive.ml",_)`

Conclusions

- Syntactic criteria are dead: Gimenez, Blanqui, Barthe (and...) moved to type-based guard verification (size annotation)

Towards type based termination...

Which system ?

- CIC sombrero ? (Jorge)
- implicit product over ordinals ? (Andreas)

Set-theoretical semantics ?

More challenges

Example

```
Fixpoint minus a b :=  
  match a,b with 0, _ => 0 | ... end
```

vs.

```
Fixpoint minus a b :=  
  match a,b with 0, _ => a | ... end
```

Overview

- 1 Theory
 - Basic criterion
 - Extensions
- 2 Algorithm
 - Efficiency
- 3 Discussion
- 4 Attic

Positivity condition (since you insist)

- Crucial for consistency
- Lists

```
Inductive list (A:Type) : Type :=
  nil | cons (x:A) (l:list A).
```

- Ordinals `Inductive ord:Set := 0 | S(o:ord) | lim(f:nat→ord).`
- Useful extension: nested inductive types
`Inductive tree:Set := Node(l:list tree).`
 Reuse list library

Positivity condition (since you insist)

Definition (strict positivity)

$\Pi \vec{x} : \vec{t}. C$ is strictly positive w.r.t. X if forall i either:

- (Norec) X does not occur free in t_i , or
- (Rec) $t_i = \Pi \vec{y} : \vec{u}. X \vec{w}$ where X does not occur in $\vec{u}\vec{w}$, or
- (Nested) $t_i = \Pi \vec{y} : \vec{u}. \text{Ind}(Y : B)\{\vec{D}\} \vec{w}$ and
 - X does not occur free in $\vec{u}\vec{w}$
 - D_i is strictly positive w.r.t. X forall i

Impredicativity

Recursive calls cannot be allowed on *all* constructor arguments

```
Inductive I : Set := C (f:forall A:Set,A->A).  
Fixpoint F (x:I) : False :=  
  match x with  
  | C f => F (f I x)  
  end
```

Definition (recursive positions)

constructors arguments that satisfy (Rec) or (Nested) clause of positivity.

Definition of the condition (boring cases)

Simply check recursively that subexpressions are guarded

$$\frac{f \notin FV(M)}{M \in \text{Norm}_{\rho}^{f,k}} \quad \frac{T \in \text{Norm}_{\rho}^{f,k} \quad U \in \text{Norm}_{\rho}^{f,k}}{\Pi x : T U \in \text{Norm}_{\rho}^{f,k}}$$

$$\frac{T \in \text{Norm}_{\rho}^{f,k} \quad U \in \text{Norm}_{\rho}^{f,k}}{\lambda x : T U \in \text{Norm}_{\rho}^{f,k}} \quad \frac{M \in \text{Norm}_{\rho}^{f,k} \quad N \in \text{Norm}_{\rho}^{f,k}}{M N \in \text{Norm}_{\rho}^{f,k}}$$

Nested vs. mutual inductive types

Example (Guard violated)

```
Fixpoint size (t:tree) :=
  match t with
  | Node l => S(size_forest l)
  | _ => 0
  end
with size_forest (l:list tree) :=
  match l with
  | nil => 0
  | t::l' => size t + size l'
  end.
```

Mutual inductive types can be used in the context of both mutual fixpoints and nested fixpoints.

Nested inductive types cannot be used in the context of mutual fixpoints.