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

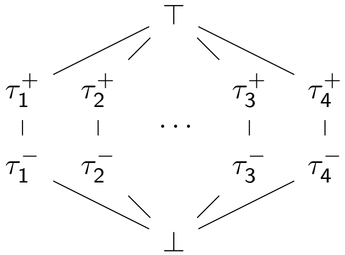
  Translation towards recursors:

  For $f : I \to T$, define I_f similar to I such that every subterm of type I comes with its image by f. Then write $g : I \to I_f$ and $h : I_f \to T$.
- Blanqui (05), Calculus of Algebraic Constructions: reducibility proof (CC + higher order rewriting)
- Only works for the basic criterion.

Basic criterion: size information

$$
\begin{aligned}
\text{(non-strict)}\quad & \Sigma^{+} ::= \top \mid \tau^{+} & (\top = \text{not a subterm})\\
\text{(strict)}\quad & \Sigma^{-} ::= \bot \mid \tau^{-} & (\bot = \text{inaccessible cases})\\
\text{(size info)}\quad & \Sigma ::= \Sigma^{+} \cup \Sigma^{-} &
\end{aligned}
$$

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 $\mathrm{Str}(I, \vec{C})$
  (not any constructor argument because of impredicativity)

Lemma

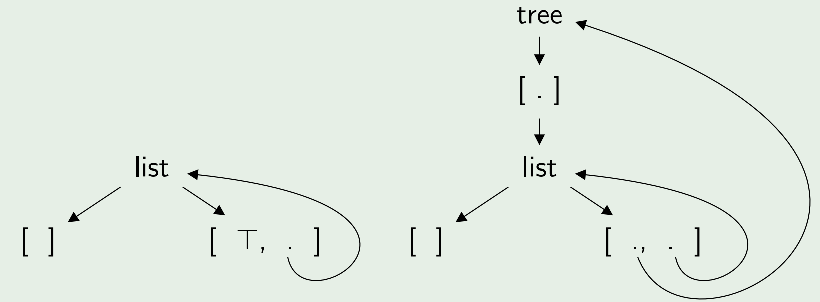
The computed tree is the set of paths that cannot contain an infinite number of inductive objects.

Regular trees: examples

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

Example ($\mathrm{Str}($*list*$)$ and $\mathrm{Str}($*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 $M \in \mathrm{Norm}_\rho^{F,k}$ meaning that M does recursive calls to F only on strict subterms, as specified by ρ
- Pattern-matching propagates information on pattern variables

$$\mathrm{Constr}(i, I)\ x_1 \ldots x_k \mid \sigma = \{(x_j, \sigma.i.j^-) \mid j \leq k\}$$

Remarks

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

Definition of the condition (1)

Definition (Terms)

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

Typing rule:

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

Initializing the termination check ($\mathrm{Str}()$ precomputed):

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

Definition of the condition (2)

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

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

+ congruence rules...

Subterms

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

Checking guard modulo reduction

- Enable reusability of standard definitions

In fact, the typing rule for fixpoints is:

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

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 inaccessibles cases (absurd) elimination

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

Example

```
Definiton 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

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

Example

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

Nested fixpoints

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

Example (size of a tree)

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

Overview

Implementation

Follows the (almost) syntax-directed rules presented here

- `subterm_specif` implements $\rho \vdash^{S} M \Rightarrow \sigma$
- `check_one_fix` implements $M \in \mathrm{Norm}_{\rho}^{f,k}$
  - 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 := id (id (id F)) (pred x).
```

Propagation of size through match

- size of pattern variable is computed eagerly

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

Issue:

- Need to reduce terms to get the most precise information on match subject, but it is not used!
  `Fixpoint F x := if <long comp> then F (pred x)`

Overview

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 criterions 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

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 :=`

  ```
     O | 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) $t_i = \Pi\vec{y} : \vec{u}.\, \mathrm{Ind}(Y : B)\{\vec{D}\}\ \vec{w}$ 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

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

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

Nested vs. mutual inductive types

Example (Guard violated)

```
Fixpoint size (t:tree) :=
  match t with
      Node l ⇒ S(size_forest l)
  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.