

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

# Terminaison de fonctions récurives pour Focal

William Bartlett

ENSIIE – Équipe CPR, Cédric

19 juin 2007

Terminaison  
de fonctions  
récur­sives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récur­sive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

Contexte : certification du code Focal par Coq.

## Correction totale

fonctions récur­sives  $\rightsquigarrow$  preuve de terminaison.

Preuve de terminaison :

- ordre bien fondé,
- appel récur­sif  $\Rightarrow$  preuve de décroissance.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

Contexte : certification du code Focal par Coq.

## Correction totale

fonctions récurives  $\rightsquigarrow$  preuve de terminaison.

Preuve de terminaison :

- ordre bien fondé,
- appel récurif  $\Rightarrow$  preuve de décroissance.

- Récursion structurelle : triviale.

```
let rec fact = function  
| Zero -> Succ Zero  
| Succ n -> mult_nat(Succ n, fact n);;
```

$$\forall n \in \mathbb{N}, \quad n \prec \text{Succ } n$$

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

- Récursion structurelle : triviale.

```
let rec fact = function  
| Zero -> Succ Zero  
| Succ n -> mult_nat(Succ n, fact n);;
```

$$\forall n \in \mathbb{N}, \quad n \prec \text{Succ } n$$

- Récur­sion non structurelle : plus délicate.

```
let rec ack = function
```

```
| (Zero, n) -> Succ n
```

```
| (Succ m, Zero) -> ack(m, Succ Zero)
```

```
| (Succ m, Succ n) -> ack(m, ack(Succ m, n))
```

$$\forall (m, n, p) \in \mathbb{N}^3, \quad (m, n) \prec (\text{Succ } m, p)$$

$$\forall (m, n) \in \mathbb{N}^2, \quad (m, n) \prec (m, \text{Succ } n)$$

- Récur­sion non structurelle : plus délicate.

```
let rec ack = function
```

```
| (Zero, n) -> Succ n
```

```
| (Succ m, Zero) -> ack(m, Succ Zero)
```

```
| (Succ m, Succ n) -> ack(m, ack(Succ m, n))
```

$$\forall (m, n, p) \in \mathbb{N}^3, \quad (m, n) \prec (\text{Succ } m, p)$$

$$\forall (m, n) \in \mathbb{N}^2, \quad (m, n) \prec (m, \text{Succ } n)$$

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

## Cas de Focal

Récursion bien fondée mal gérée.

Traduction en Coq : preuve de terminaison incorrecte.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

Élargir l'espace des fonctions correctement compilées par Focal.

- Modifier le compilateur vers Coq :
  - proposer un schéma plus universel,
  - profiter des avancées en Coq sur le traitement des fonctions récurives.
- Étendre la syntaxe de Focal pour faciliter cette compilation.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

1 Existant

2 Solution

3 Conséquences

4 Conclusion

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

## Existant

Définition d'une  
fonction  
récurive  
Compilation

## Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

## Conséquences

Dépendances

## Conclusion

- 1 Existant
  - Définition d'une fonction récurive
  - Compilation
- 2 Solution
  - Extension de syntaxe
  - Ajout de la bonne fondation
  - Exemples d'utilisation
  - Compilation
- 3 Conséquences
  - Dépendances
- 4 Conclusion

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récursive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

```
let rec gcd(a in self , b in self) =  
  if !is_zero(b)  
  then a  
  else let r = !remainder(a,b) in  
    if !is_zero(r)  
    then b  
    else !gcd(b,r);
```

## Section Gcd.

**Variable** `abst_T` : `Set`.

**Variable** `abst_remainder` :

`abst_T`  $\rightarrow$  `abst_T`  $\rightarrow$  `abst_T`.

**Variable** `is_zero` : `abst_T`  $\rightarrow$  `bool__t`.

```
Let my_order: ((prod (abst_T) (abst_T)))  $\rightarrow$ 
  ((prod (abst_T) (abst_T)))  $\rightarrow$  Prop :=
  (fun(x: (prod (abst_T) (abst_T)))  $\Rightarrow$ 
    (fun(y: (prod (abst_T) (abst_T)))  $\Rightarrow$ 
      (magic_order x y))).
```

**Let** `well_founded_my_order`:

(`well_founded` `my_order`).

`eapply` `dontwanttoproveit`.

**Qed**.

## Section Gcd.

**Variable** `abst_T` : `Set`.

**Variable** `abst_remainder` :

`abst_T`  $\rightarrow$  `abst_T`  $\rightarrow$  `abst_T`.

**Variable** `is_zero` : `abst_T`  $\rightarrow$  `bool__t`.

**Let** `my_order`: ((`prod` (`abst_T`) (`abst_T`)))  $\rightarrow$   
 ((`prod` (`abst_T`) (`abst_T`)))  $\rightarrow$  `Prop` :=  
 (**fun** (`x`: (`prod` (`abst_T`) (`abst_T`)))  $\Rightarrow$   
 (**fun** (`y`: (`prod` (`abst_T`) (`abst_T`)))  $\Rightarrow$   
 (`magic_order` `x` `y`))).

**Let** `well_founded_my_order`:

(`well_founded` `my_order`).

**eapply** `dontwanttoproveit`.

**Qed**.

```

Let lemma_1:
  ∀ _aux1:(prod (abst_T) (abst_T)),
  let x:=(__g_first _aux1) in
  let _aux2:=(__g_scnd _aux1) in
  let y:=_aux2 in
  (Is_true (__g_not_b (abst_is_zero y))) →
  let r:=(abst_remainder x y) in
  (Is_true (__g_not_b (abst_is_zero r))) →
  unit → (my_order (y, r) _aux1).
  eapply dontwanttoproveit.
Qed.

```

Terminaison  
de fonctions  
récur­sives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récur­sive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépé­dances

Conclusion

```

Let gcd_aux:  $\forall$  _aux1: (prod (abst_T) (abst_T)),
  ( $\forall$  _y:(prod (abst_T) (abst_T)),
  (my_order _y _aux1)  $\rightarrow$  abst_T)  $\rightarrow$  abst_T :=
fun _aux1: (prod (abst_T) (abst_T))  $\Rightarrow$ 
fun gcd_aux_rec: ( $\forall$  _y:(prod(abst_T)(abst_T)),
  (my_order _y _aux1)  $\rightarrow$  abst_T)  $\Rightarrow$ 
let x:= (__g_first _aux1) in
let _aux2:= (__g_scnd _aux1) in
let y:= _aux2 in
if (abst_is_zero y) then x
else let r:= (abst_remainder x y ) in
  if (abst_is_zero r) then y
  else
    (gcd_aux_rec (y,r) (lemma_1 _aux1
  (dontwanttoproveit (Is_true (__g_not_b
  (abst_is_zero y)))) (dontwanttoproveit
  (Is_true(__g_not_b(abst_is_zero r))))))tt)).
  
```

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

```
Let euclidean_semi_ring_gcd_aux:=  
  (Fix well_founded_my_order  
  (fun(x: (prod (abst_T) (abst_T)))  $\Rightarrow$   
    abst_T) gcd_aux).
```

```
Definition euclidean_semi_ring__gcd:=  
  fun x y  $\Rightarrow$  (euclidean_semi_ring_gcd_aux  
    (pair x y)).
```

```
End Gcd.
```

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

```
Let euclidean_semi_ring_gcd_aux:=  
  (Fix well_founded_my_order  
  (fun(x: (prod (abst_T) (abst_T)))  $\Rightarrow$   
    abst_T) gcd_aux).
```

```
Definition euclidean_semi_ring__gcd:=  
  fun x y  $\Rightarrow$  (euclidean_semi_ring_gcd_aux  
    (pair x y)).
```

```
End Gcd.
```

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive

Compilation

Solution

Extension de  
syntaxe

Ajout de la  
bonne fondation

Exemples  
d'utilisation

Compilation

Conséquences

Dépendances

Conclusion

- La fonction et sa preuve séparées du reste.
- Un ordre bien fondé.
- Des lemmes de décroissance.

Mais

- Absence de vraies preuves.
- Ordre non implanté.

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive

Compilation

Solution

Extension de  
syntaxe

Ajout de la  
bonne fondation

Exemples  
d'utilisation

Compilation

Conséquences

Dépendances

Conclusion

- La fonction et sa preuve séparées du reste.
- Un ordre bien fondé.
- Des lemmes de décroissance.

Mais

- **Absence** de vraies preuves.
- Ordre **non implanté**.

# Conclusion : Que doit-on faire ?

- Inclure la preuve dans le fichier source Focal.
- Conserver les mécanismes :
  - de retard de preuve,
  - des preuves Zenon.
- Profiter de **Function** [Coq v8.1].

```
Function f (x1:t1) ... (xn:tn)
{wf R xi} : t = ...
```

```
Proof .
```

```
...
Qed .
```

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- 1 Existant
  - Définition d'une fonction récurive
  - Compilation
- 2 Solution
  - Extension de syntaxe
  - Ajout de la bonne fondation
  - Exemples d'utilisation
  - Compilation
- 3 Conséquences
  - Dépendances
- 4 Conclusion

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Basée sur :
  - existant,
  - **Function**,
  - syntaxe des preuves de théorèmes.
  
- Nouvelles *indications* de preuve :
  - **structural**,
  - **wforder**.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
réursive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation

Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

- Réursion structurelle :
  - argument de type inductif.
- Réursion non structurelle :
  - ordre,
  - argument qui décroît,
  - preuve
    - ordre bien fondé,
    - appel récursif  $\Rightarrow$  argument décroissant.

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récursive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

## ■ Récursion structurelle

```
let rec f(x1 in t1, ... xn in tn) in t = ...
proof: structural xi;
```

## ■ Récursion non structurelle :

```
let rec f(x1 in t1, ... xn in tn) in t = ...
proof: <1>1 wforder R xi
...;
```

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récurive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

## Accessibilité

Un élément est  $R$ -accessible ssi tout élément plus petit est  $R$ -accessible.

## Bonne fondation

Un ordre  $R$  sur  $E$  est bien fondé ssi tout élément de  $E$  est  $R$ -accessible.

Accessibilité n'est pas typable en Focal.

d'où import depuis Coq.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récurive  
Compilation

Solution  
Extension de  
syntaxe  
**Ajout de la  
bonne fondation**

Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

## En général

- Montrer qu'un ordre est bien fondé :
  - directement,
  - à partir d'un autre ordre bien fondé,
  - à partir d'une mesure.

## Ajoutés à Focal

- Ordres bien fondés basiques.
- Théorèmes sur la bonne fondation.

## En général

- Montrer qu'un ordre est bien fondé :
  - directement,
  - à partir d'un autre ordre bien fondé,
  - à partir d'une mesure.

## Ajoutés à Focal

- Ordres bien fondés basiques.
- Théorèmes sur la bonne fondation.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation

Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

```
let rec length(l in list('a)) in int =  
match l with  
| #Nil → 0  
| #Cons(_, t) → #int_plus(length(t), 1)  
end;
```

# Exemples d'utilisation : Exemple 1

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récurive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation

Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

```
let rec length(l in list('a)) in int =  
match l with  
| #Nil → 0  
| #Cons(_, t) → #int_plus(length(t), 1)  
end  
proof: structural l;
```

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation

Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

```
let rec gcd(a in self , b in self) =  
  if !is_zero(b)  
  then a  
  else let r = !remainder(a,b) in  
    if !is_zero(r)  
    then b  
    else !gcd(b,r);
```

## Exemples d'utilisation : Exemple 2

```

let rec gcd(a in self , b in self) =
  if !is_zero(b)
  then a
  else let r = !remainder(a,b) in
    if !is_zero(r)
    then b
    else !gcd(b,r)

```

proof:

```

<1>1 wforder !euclidean_order b
<2>1
  (* bonne fondation de *
  * !euclidean_order *)
<2>2
  (* décroissance de *
  * l'argument b *)
<2>f qed by <2>1,<2>2
<1>f qed;

```

Terminaison  
de fonctions  
récur­sives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récur­sive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation

Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

## Section Gcd.

**Variable** wforder: relation(self\_T \* self\_T).

**Variable** termination\_gcd: ... .

**Function** gcd\_aux (\_\_arg:self\_T \* self\_T): self\_T  
{**wf** wforder \_\_arg} := **let** (a,b) := \_\_arg **in**  
**corps de fonction**

**Proof.** intuition. intuition. **Qed.**

**Definition** gcd := **fun** a b: self\_T =>  
gcd\_aux (a,b)

**End** Gcd.

**Definition** s\_wforder\_gcd := wforder  
(self\_T \* self\_T) 1 s\_euclidean\_order.

**Lemma** s\_termination\_gcd : ... .  
**preuve**

**Qed.**

**Let** s\_gcd = gcd s\_wforder\_gcd  
s\_termination\_gcd.

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- 1 Existant
  - Définition d'une fonction récursive
  - Compilation
- 2 Solution
  - Extension de syntaxe
  - Ajout de la bonne fondation
  - Exemples d'utilisation
  - Compilation
- 3 Conséquences
  - Dépendances
- 4 Conclusion

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

**Conséquences**

Dépendances

Conclusion

## ■ Possibilités :

- retard de la preuve
- preuve en Zenon

## ■ Limites :

- définitions mutuellement récurives,
- fonctions récurives locales.

## ■ Nouvelles dépendances.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récurive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

**Conséquences**

Dépendances

Conclusion

## ■ Possibilités :

- retard de la preuve
- preuve en Zenon

## ■ Limites :

- définitions mutuellement récurives,
- fonctions récurives locales.

## ■ Nouvelles dépendances.

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Possibilités :
  - retard de la preuve
  - preuve en Zenon
- Limites :
  - définitions mutuellement récurives,
  - fonctions récurives locales.
- Nouvelles dépendances.

## Dépendances entre preuve et fonction

Soit  $s$  une espèce. Soient  $f$  une fonction et  $p$  une propriété définies dans  $s$ .

- Si la preuve de  $p$  dépend de l'existence de  $f$ , alors  $p$  **decl-dépend** de  $f$ .
- Si la preuve de  $p$  dépend de l'implantation de  $f$ , alors  $p$  **def-dépend** de  $f$ .

<b>theorem</b> $p : P[f]$	<b>theorem</b> $p : P[f]$
<b>proof</b> : ... ;	<b>proof</b> : <b>def</b> $f$ ... ;

## Propriété

Soit  $s$  une espèce. Soient  $f$  une fonction et  $p$  une propriété définies dans  $s$ , avec  $p$  def-dépendant de  $f$ .

Toute sous-espèce de  $s$  qui redéfinit  $f$  doit contenir une nouvelle preuve de  $p$ .

**proof** of  $p = \dots$  ;

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Nouvelles dépendances :
  - def-dépendance entre l'**énoncé** à prouver et la fonction réursive,
  - decl-dépendance entre la fonction réursive, et sa preuve de terminaison,
  - éventuelles dépendances entre la preuve de terminaison et d'autres fonctions.

## Enoncé à prouver

Species  $s_1$ 

```
let rec f(x in t) = ... f(g(x));
```

$$\forall x \in t, \quad g(x) < x$$
Species  $s_2$  inherits  $s_1$ 

```
let rec f(x in t) = ... f(g1(x)) + f(g2(x));
```

$$\forall x \in t, \quad g_1(x) < x \wedge g_2(x) < x$$
Species  $s_3$  inherits  $s_1$ 

```
let f(x in t) = ...;
```

Terminaison  
de fonctions  
récur­sives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récur­sive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

# Enoncé à prouver

## Species $s_1$

```
let rec f(x in t) = ... f(g(x));
```

$$\forall x \in t, \quad g(x) < x$$

## Species $s_2$ inherits $s_1$

```
let rec f(x in t) = ... f(g1(x)) + f(g2(x));
```

$$\forall x \in t, \quad g_1(x) < x \wedge g_2(x) < x$$

## Species $s_3$ inherits $s_1$

```
let f(x in t) = ...;
```

Terminaison  
de fonctions  
récur­sives  
pour Focal

William  
Bartlett

Existant  
Définition d'une  
fonction  
récur­sive  
Compilation

Solution  
Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences  
Dépendances

Conclusion

# Enoncé à prouver

## Species $s_1$

```
let rec f(x in t) = ... f(g(x));
```

$$\forall x \in t, \quad g(x) < x$$

## Species $s_2$ inherits $s_1$

```
let rec f(x in t) = ... f(g1(x)) + f(g2(x));
```

$$\forall x \in t, \quad g_1(x) < x \wedge g_2(x) < x$$

## Species $s_3$ inherits $s_1$

```
let f(x in t) = ...;
```

## Dépendances de la preuve de terminaison

Soient  $r$  une fonction réursive et  $p$  sa preuve de terminaison.  
Soit  $f$  une fonction dont  $p$  def-dépend.

Que faire si  $f$  est redéfinie dans une sous-espèce ?

- `proof of termination r = ... ;`

## Dépendances de la preuve de terminaison

Soient  $r$  une fonction récurive et  $p$  sa preuve de terminaison.  
Soit  $f$  une fonction dont  $p$  def-dépend.

Que faire si  $f$  est redéfinie dans une sous-espèce ?

■ `proof of termination r = ... ;`

## Dépendances de la preuve de terminaison

Soient  $r$  une fonction réursive et  $p$  sa preuve de terminaison.  
Soit  $f$  une fonction dont  $p$  def-dépend.

Que faire si  $f$  est redéfinie dans une sous-espèce ?

- **proof of termination**  $r = \dots$  ;

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- 1 Existant
  - Définition d'une fonction récursive
  - Compilation
- 2 Solution
  - Extension de syntaxe
  - Ajout de la bonne fondation
  - Exemples d'utilisation
  - Compilation
- 3 Conséquences
  - Dépendances
- 4 Conclusion

Terminaison  
de fonctions  
récurives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
réursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Définition des fonctions récurives.
- En cours : modification du compilateur.
  - Génération des obligations de preuve ✓
  - Génération du générateur de méthode ✓
  - Ajout de commentaires au code ✓
  - Analyse de dépendances
  - Résolution de l'héritage
- À terme : mise à disposition d'un solveur de terminaison de fonctions / systèmes de réécriture.

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Définition des fonctions récursives.
- En cours : modification du compilateur.
  - Génération des obligations de preuve ✓
  - Génération du générateur de méthode ✓
  - Ajout de commentaires au code ✓
  - Analyse de dépendances
  - Résolution de l'héritage
- À terme : mise à disposition d'un solveur de terminaison de fonctions / systèmes de réécriture.

Terminaison  
de fonctions  
récursives  
pour Focal

William  
Bartlett

Existant

Définition d'une  
fonction  
récursive  
Compilation

Solution

Extension de  
syntaxe  
Ajout de la  
bonne fondation  
Exemples  
d'utilisation  
Compilation

Conséquences

Dépendances

Conclusion

- Définition des fonctions récursives.
- En cours : modification du compilateur.
  - Génération des obligations de preuve ✓
  - Génération du générateur de méthode ✓
  - Ajout de commentaires au code ✓
  - Analyse de dépendances
  - Résolution de l'héritage
- À terme : mise à disposition d'un solveur de terminaison de fonctions / systèmes de réécriture.