



# Spécification et analyse de programmes C(++) avec Frama-C et ACSL

Virgile Prevosto

CEA List, Saclay

25 septembre 2007

# Plan de la présentation



## Frama-C

Une plate-forme d'analyse de programmes C  
Frama-C++



## ACSL

Présentation  
Exemple



## Modélisation d'un programme C++

Contexte  
Définition d'un modèle purement logique  
Définition d'un modèle ghost



## Frama-C

Une plate-forme d'analyse de programmes C  
Frama-C++



## ACSL

Présentation  
Exemple



## Modélisation d'un programme C++

Contexte  
Définition d'un modèle purement logique  
Définition d'un modèle ghost

# Présentation

## Contexte

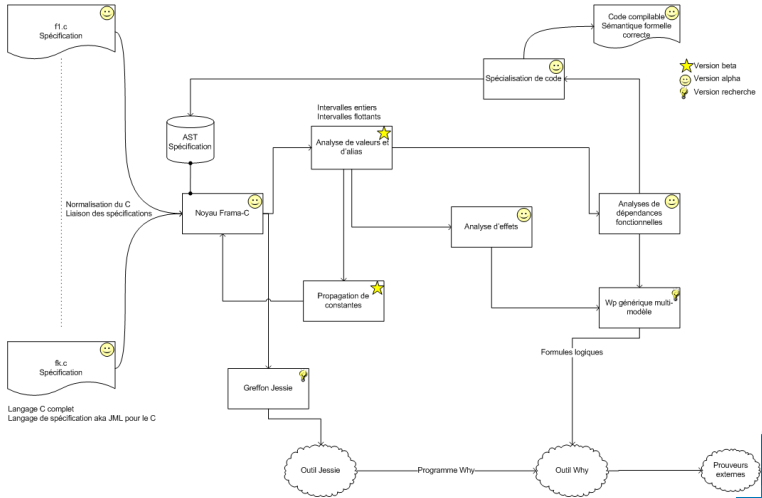
- Projet RNTL CAT
- Développé au CEA LIST (LSL) et à l'université Paris-Sud (LRI)
- "Successeur" de CAVEAT et Caduceus

## Principes

- *Framework for Modular Analyses of C*
- noyau basé sur CIL (Berkeley)
- greffons d'analyse pouvant coopérer entre eux
- langage d'annotations logiques ACSL
- annotations spécialisées pour certains greffons



# Architecture



# Extension au C++

- Utilisation du système Frama-C pour analyser du C++
- Nouveau(x) greffon(s) dédiés
- Traduction C++ vers C
- Front-end existant: Elsa (Berkeley)



# Constructions supportées

- C++ “de base”
  - constructeurs/destructeurs
  - variables temporaires
  - références
- templates (ceux qu'Elsa sait instantier)
- conversions utilisateurs
- divers points mineurs



# Constructions supportées

- C++ “de base”
  - constructeurs/destructeurs
  - variables temporaires
  - références
- templates (ceux qu'Elsa sait instantier)
- conversions utilisateurs
- divers points mineurs

## Principaux points non traités

- héritage



# Constructions supportées

- C++ “de base”
  - constructeurs/destructeurs
  - variables temporaires
  - références
- templates (ceux qu'Elsa sait instantier)
- conversions utilisateurs
- divers points mineurs

## Principaux points non traités

- héritage
- exceptions





## Frama-C

Une plate-forme d'analyse de programmes C  
Frama-C++



## ACSL

Présentation  
Exemple



## Modélisation d'un programme C++

Contexte  
Définition d'un modèle purement logique  
Définition d'un modèle ghost

# Spécifier un programme C

- Langage de spécification à la JML
- Orienté analyse et vérification **statique**
- Version de travail interne à CAT en juillet 2007
- Version publique en janvier 2008
- Parseur en cours de développement



# Objectifs

## Fondamentaux

- Parler avec précision des objets C
- Sûreté de fonctionnement, terminaison



# Objectifs



## Fondamentaux

- Parler avec précision des objets C
- Sûreté de fonctionnement, terminaison

## Avancés

- Échange d'information entre analyses

# Objectifs



## Fondamentaux

- Parler avec précision des objets C
- Sûreté de fonctionnement, terminaison

## Avancés

- Échange d'information entre analyses
- Modélisation, spécification formelle

# Constructions supportées

## Annotations du code C

- Comportements des fonctions
- Assertions dans le code



# Constructions supportées

## Annotations du code C

- Comportements des fonctions
- Assertions dans le code

## Spécifications algébriques

- prédicats fonctions logiques
- définitions axiomatiques



# Constructions supportées

## Annotations du code C

- Comportements des fonctions
- Assertions dans le code

## Spécifications algébriques

- prédicats fonctions logiques
- définitions axiomatiques

## Code “ghost”.

- instrumentation du code original
- “spécification exécutable”
- sans modifier la sémantique du programme



# Constructions supportées

## Annotations du code C

- Comportements des fonctions
- Assertions dans le code

## Spécifications algébriques


- prédicats fonctions logiques
- définitions axiomatiques

## Code “ghost”.

- instrumentation du code original
- “spécification exécutable”
- **sans modifier la sémantique du programme**




## Exemple




```
int max(int t[], int n) {
    int imax = 0, i;
    if (n<=0) return -1;
    for(i = 1; i < n; i++) {
        if (t[i] > t[imax]) {
            imax = i;
        }
    }
    return imax;
}
```

## Exemple



```
/*@ requires \valid_range(t,0,n-1);
   behavior nonempty:
     assumes n > 0;
     ensures 0<= \result < n &&
        (\forallall int i; 0 <= i < n ==>
           t[\result] >= t[i]);
   behavior empty:
     assumes n <= 0;
     ensures \result == -1;
*/
int max(int t[], int n) {
```

## Exemple




```
int max(int t[], int n) {
    int imax = 0, i;
    if (n<=0) return -1;
    /*@ loop invariant
       (\forall int j; 0<= j < i ==>
        t[imax] >= t[j]);
    */
    for(i = 1; i < n; i++) {
        if (t[i] > t[imax]) {
            imax = i;
        }
    }
    return imax;
}
```

## Exemple

```
int max(int t[], int n) {
    int imax = 0, i;
    /*@ ghost int max; */
    if (n<=0) return -1;
    /*@ ghost max = t[0]; */
    /*@ loop invariant
       (\forall int j; 0<= j < i ==>
        max >= t[j]);
    */
    for(i = 1; i < n; i++) {
        if (t[i] > t[imax]) {
            imax = i;
            /*@ ghost max = t[i]; */
        }
    }
    return imax;
}
```



## Exemple



```
/*@ predicate is_max(int max, int t[], int len); */
/*@ axiom max_gt:
\forall int max, int t[], int len, int i;
is_max(max,t,len) ==> 0 <= i < len ==> t[i] <= max;
*/
/*@ axiom max_eq: ... */
/*@ requires \valid_range(t,0,n-1);
behavior nonempty:
    assumes n > 0;
    ensures 0<= \result < n &&
           is_max(t[\result],t,n);
*/
int max(int t[], int n) {
```

## Exemple

```
int max(int t[], int n) {
    int imax = 0, i;
    /*@ ghost int max; */
    if (n<=0) return -1;
    /*@ loop invariant
       is_max(max,t,i-1);
    */
    for(i = 1; i < n; i++) {
        if (t[i] > t[imax]) {
            imax = i;
        }
    }
    return imax;
}
```





## Frama-C

Une plate-forme d'analyse de programmes C  
Frama-C++



## ACSL

Présentation  
Exemple



## Modélisation d'un programme C++

Contexte  
Définition d'un modèle purement logique  
Définition d'un modèle ghost

# Pistachio

- Micro-noyau de type L4
- C++ “simple”
- Spécification en langue naturelle des appels systèmes
- But: vérifier la correction de l’implantation ...
- ... pour chaque comportement possible d’un appel




## Exemple - sys\_schedule

```
SYS_SCHEDULE (threadid_t dest_tid,  
              word_t time_control, word_t processor_control,  
              word_t prio, word_t preemption_control ) { ... }
```



## Exemple - sys\_schedule



```
SYS_SCHEDULE (threadid_t dest_tid,  
              word_t time_control, word_t processor_control,  
              word_t prio, word_t preemption_control ) { ... }
```

*The system call can be used by schedulers to define the priority, timeslice length, and other scheduling parameters [...]*  
*The system call is only effective if the calling thread is defined as the destination thread's scheduler*

## Exemple - sys\_schedule

```
SYS_SCHEDULE (threadid_t dest_tid,  
              word_t time_control, word_t processor_control,  
              word_t prio, word_t preemption_control ) { ... }
```

*The system call can be used by schedulers to define the priority, timeslice length, and other scheduling parameters [...]  
The system call is only effective if the calling thread is defined as the destination thread's scheduler*

dest	Destination thread ID. The destination thread must be existent (but can be inactive)
	All further input parameters have no effect if the supplied value is -1 [...]
prio	New priority for destination thread. Must be less than or equal to current thread



## Exemple - sys\_schedule

```
SYS_SCHEDULE (threadid_t dest_tid,  
              word_t time_control, word_t processor_control,  
              word_t prio, word_t preemption_control ) { ... }
```

*The system call can be used by schedulers to define the priority, timeslice length, and other scheduling parameters [...]*

***The system call is only effective if the calling thread is defined as the destination thread's scheduler***

dest	Destination thread ID. <b>The destination thread must be existent</b> (but can be inactive)
	All further input parameters have no effect if the supplied value is -1 [...]
prio	New priority for destination thread. <b>Must be less than or equal to current thread</b>



# Définition d'un modèle purement logique

- types logiques et fonctions logiques abstraites
- axiomatisation de leurs effets
- prédicats et fonctions de correspondance dans les classes concrètes
- comportement des fonctions concrètes spécifié à partir des fonctions abstraites



## Exemple

### tcb logique

```
/*@ logic type logic_tcb; */
/*@ predicate existing(logic_tcb t) reads t; */
/*@ logic logic_tcb
    logic_scheduler(logic_tcb t) reads t; */
/*@ logic logic_tcb
    L4_schedule_prio(logic_tcb t, prio_t p) reads t,p; */
/*@ axiom L4_schedule_prio_prio:
    \forall logic_tcb t; \forall prio_t p;
    (logic_scheduler(t)==current_tcb.logic_bound() &&
     logic_priority(current_tcb.logic_bound()) > p &&
     p >= 0 && p <= MAX_PRIO) ==>
    logic_priority(L4_schedule_prio(t,p)) == p;
*/
```



# Exemple

## tcb logique

```
/*@ logic type logic_tcb; */
/*@ predicate existing(logic_tcb t) reads t; */
/*@ logic logic_tcb
    logic_scheduler(logic_tcb t) reads t; */
/*@ logic logic_tcb
    L4_schedule_prio(logic_tcb t, prio_t p) reads t,p; */
/*@ axiom L4_schedule_prio_prio:
    \forall logic_tcb t; \forall prio_t p;
    (logic_scheduler(t)==current_tcb.logic_bound() &&
     logic_priority(current_tcb.logic_bound()) > p &&
     p >= 0 && p <= MAX_PRIO) ==>
    logic_priority(L4_schedule_prio(t,p)) == p;
*/
```



## Exemple

### tcb logique

```
/*@ logic type logic_tcb; */
/*@ predicate existing(logic_tcb t) reads t; */
/*@ logic logic_tcb
    logic_scheduler(logic_tcb t) reads t; */
/*@ logic logic_tcb
    L4_schedule_prio(logic_tcb t, prio_t p) reads t,p; */
/*@ axiom L4_schedule_prio_prio:
    \forall logic_tcb t; \forall prio_t p;
    (logic_scheduler(t)==current_tcb.logic_bound() &&
     logic_priority(current_tcb.logic_bound()) > p &&
     p >= 0 && p <= MAX_PRIO) ==>
    logic_priority(L4_schedule_prio(t,p)) == p;
*/
```



## Exemple

### tcb logique

```
/*@ logic type logic_tcb; */
/*@ predicate existing(logic_tcb t) reads t; */
/*@ logic logic_tcb
    logic_scheduler(logic_tcb t) reads t; */
/*@ logic logic_tcb
    L4_schedule_prio(logic_tcb t, prio_t p) reads t,p; */
/*@ axiom L4_schedule_prio_prio:
    \forall logic_tcb t; \forall prio_t p;
    (logic_scheduler(t)==current_tcb.logic_bound() &&
     logic_priority(current_tcb.logic_bound()) > p &&
     p >= 0 && p <= MAX_PRIO) ==>
    logic_priority(L4_schedule_prio(t,p)) == p;
*/
```



## Exemple

### tcb logique

```
/*@ logic type logic_tcb; */
/*@ predicate existing(logic_tcb t) reads t; */
/*@ logic logic_tcb
    logic_scheduler(logic_tcb t) reads t; */
/*@ logic logic_tcb
    L4_schedule_prio(logic_tcb t, prio_t p) reads t,p; */
/*@ axiom L4_schedule_prio_prio:
    \forall logic_tcb t; \forall prio_t p;
    (logic_scheduler(t)==current_tcb.logic_bound() &&
     logic_priority(current_tcb.logic_bound()) > p &&
     p >= 0 && p <= MAX_PRIO) ==>
    logic_priority(L4_schedule_prio(t,p)) == p;
*/
```



## Exemple (suite)

### Lien avec l'implantation

```
class tcb_t ...
/*@ logic logic_tcb logic_bound(); */
/*@ predicate is_bound(logic_tcb t)
    { myself_global == logic_myself_global(t) &&
      logic_existing() <==> existing(t) &&
      priority == logic_priority(t) &&
      logic_get_tcb(scheduler)->logic_bound() ==
        logic_scheduler(t)
    }
*/
/*@ class invariant model:
    is_bound(logic_bound()); */
```



## Exemple (suite)

### Lien avec l'implantation

```
class tcb_t ...
/*@ logic logic_tcb logic_bound(); */
/*@ predicate is_bound(logic_tcb t)
    { myself_global == logic_myself_global(t) &&
      logic_existing() <==> existing(t) &&
      priority == logic_priority(t) &&
      logic_get_tcb(scheduler)->logic_bound() ==
        logic_scheduler(t)
    }
*/
/*@ class invariant model:
    is_bound(logic_bound()); */
```



## Exemple (suite)

### Lien avec l'implantation

```
class tcb_t ...
/*@ logic logic_tcb logic_bound(); */
/*@ predicate is_bound(logic_tcb t)
    { myself_global == logic_myself_global(t) &&
      logic_existing() <==> existing(t) &&
      priority == logic_priority(t) &&
      logic_get_tcb(scheduler)->logic_bound() ==
        logic_scheduler(t)
    }
*/
/*@ class invariant model:
    is_bound(logic_bound()); */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ requires
    existing(logic_global_tcb(dest_tid)) &&
    current_tcb.logic_bound() ==
    logic_scheduler(logic_global_tcb(dest_tid));
behavior schedule_prio:
    assumes prio != -1;
    requires prio >= 0 && prio <= MAX_PRIO &&
    logic_priority(current_tcb.logic_bound()) > prio;
    ensures
    logic_priority(L4_schedule_prio
        (\old(logic_global_tcb(dest_tid)),prio)) ==
    logic_priority(logic_global_tcb(dest_tid)); */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ requires
    existing(logic_global_tcb(dest_tid)) &&
    current_tcb.logic_bound() ==
    logic_scheduler(logic_global_tcb(dest_tid));
behavior schedule_prio:
    assumes prio != -1;
    requires prio >= 0 && prio <= MAX_PRIO &&
    logic_priority(current_tcb.logic_bound()) > prio;
    ensures
    logic_priority(L4_schedule_prio
        (\old(logic_global_tcb(dest_tid)),prio)) ==
    logic_priority(logic_global_tcb(dest_tid)); */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ requires
    existing(logic_global_tcb(dest_tid)) &&
    current_tcb.logic_bound() ==
    logic_scheduler(logic_global_tcb(dest_tid));
behavior schedule_prio:
    assumes prio != -1;
    requires prio >= 0 && prio <= MAX_PRIO &&
    logic_priority(current_tcb.logic_bound()) > prio;
    ensures
    logic_priority(L4_schedule_prio
        (\old(logic_global_tcb(dest_tid)),prio)) ==
    logic_priority(logic_global_tcb(dest_tid)); */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ requires
    existing(logic_global_tcb(dest_tid)) &&
    current_tcb.logic_bound() ==
    logic_scheduler(logic_global_tcb(dest_tid));
behavior schedule_prio:
    assumes prio != -1;
    requires prio >= 0 && prio <= MAX_PRIO &&
    logic_priority(current_tcb.logic_bound()) > prio;
    ensures
    logic_priority(L4_schedule_prio
        (\old(logic_global_tcb(dest_tid)),prio)) ==
    logic_priority(logic_global_tcb(dest_tid)); */
```



# Avantages et faiblesses

## Avantages

- Modélisation de haut niveau, fidèle à l'API
- Possibilité de faire des preuves sur la spec elle-même

## Inconvénients

- Impossible d'utiliser l'analyse de valeur (besoin d'un greffon de WP).
- Complexité des obligations de preuve inconnue.



## Définition d'un modèle ghost

- ensembles abstraits modélisés par des structures ghost
- membres de la structures correspondent aux fonctions abstraites
- syscalls ghost
- fonctions transformant une structure concrète en sa structure ghost
- assertions décrivant les pre et post conditions.



# Exemple

## Modèle

```
/*@ ghost struct ghost_tcb {
    bool existing; prio_t priority;
    ghost_tcb* scheduler;
    logic_global myself_global; }; */
/*@ ghost ghost_tcb ghost_map[32]; */
/*@ ghost ghost_tcb&
    ghost_global_tcb(logic_global g)
        { return ghost_map[g.raw]; }*/
/*@ ghost ghost_tcb
    ghost_schedule_prio(ghost_tcb& t,prio_t prio)
    { ghost_tcb res(t);
        res.priority = prio; return res; }*/
```



# Exemple

## Modèle

```
/*@ ghost struct ghost_tcb {
    bool existing; prio_t priority;
    ghost_tcb* scheduler;
    logic_global myself_global; }; */
/*@ ghost ghost_tcb ghost_map[32]; */
/*@ ghost ghost_tcb&
    ghost_global_tcb(logic_global g)
    { return ghost_map[g.raw]; }*/
/*@ ghost ghost_tcb
    ghost_schedule_prio(ghost_tcb& t,prio_t prio)
    { ghost_tcb res(t);
        res.priority = prio; return res; }*/
```



# Exemple

## Modèle

```
/*@ ghost struct ghost_tcb {
    bool existing; prio_t priority;
    ghost_tcb* scheduler;
    logic_global myself_global; }; */
/*@ ghost ghost_tcb ghost_map[32]; */
/*@ ghost ghost_tcb&
    ghost_global_tcb(logic_global g)
    { return ghost_map[g.raw]; }*/
/*@ ghost ghost_tcb
    ghost_schedule_prio(ghost_tcb& t, prio_t prio)
    { ghost_tcb res(t);
        res.priority = prio; return res; }*/
```



## Exemple (suite)

### Lien avec l'implantation

```
/*@ ghost ghost_tcb& make_ghost() {  
    ghost_tcb&  
    my_ghost = ghost_global_tcb(myself_global);  
    my_ghost.existing = exists();  
    my_ghost.myself_global = myself_global;  
    my_ghost.priority = priority;  
    my_ghost.scheduler = &ghost_global_tcb(scheduler);  
    return my_ghost;  
}  
*/
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ ghost ghost_tcb& dest_ghost_tcb =
        dest_tcb->make_ghost(); */
/*@ ghost ghost_tcb& current_ghost_tcb =
        current_tcb.make_ghost(); */
/*@ assert dest_ghost_tcb.scheduler ==
        &current_ghost_tcb; */
/*@ assert prio >= 0 && prio <= MAX_PRIO; */
/*@ assert current_ghost_tcb.priority > prio; */
/*@ ghost ghost_tcb& res_ghost_tcb =
        ghost_schedule_prio(dest_ghost_tcb,prio); */
... Opérations normales ...
/*@ ghost dest_ghost_tcb = dest_tcb->make_ghost(); */
/*@ assert dest_ghost_tcb == res_ghost_tcb; */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ ghost ghost_tcb& dest_ghost_tcb =
    dest_tcb->make_ghost(); */
/*@ ghost ghost_tcb& current_ghost_tcb =
    current_tcb.make_ghost(); */
/*@ assert dest_ghost_tcb.scheduler ==
    &current_ghost_tcb; */
/*@ assert prio >= 0 && prio <= MAX_PRIO; */
/*@ assert current_ghost_tcb.priority > prio; */
/*@ ghost ghost_tcb& res_ghost_tcb =
    ghost_schedule_prio(dest_ghost_tcb,prio); */
... Opérations normales ...
/*@ ghost dest_ghost_tcb = dest_tcb->make_ghost(); */
/*@ assert dest_ghost_tcb == res_ghost_tcb; */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ ghost ghost_tcb& dest_ghost_tcb =
        dest_tcb->make_ghost(); */
/*@ ghost ghost_tcb& current_ghost_tcb =
        current_tcb.make_ghost(); */
/*@ assert dest_ghost_tcb.scheduler ==
        &current_ghost_tcb; */
/*@ assert prio >= 0 && prio <= MAX_PRIO; */
/*@ assert current_ghost_tcb.priority > prio; */
/*@ ghost ghost_tcb& res_ghost_tcb =
        ghost_schedule_prio(dest_ghost_tcb,prio); */
... Opérations normales ...
/*@ ghost dest_ghost_tcb = dest_tcb->make_ghost(); */
/*@ assert dest_ghost_tcb == res_ghost_tcb; */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ ghost ghost_tcb& dest_ghost_tcb =
        dest_tcb->make_ghost(); */
/*@ ghost ghost_tcb& current_ghost_tcb =
        current_tcb.make_ghost(); */
/*@ assert dest_ghost_tcb.scheduler ==
        &current_ghost_tcb; */
/*@ assert prio >= 0 && prio <= MAX_PRIO; */
/*@ assert current_ghost_tcb.priority > prio; */
/*@ ghost ghost_tcb& res_ghost_tcb =
        ghost_schedule_prio(dest_ghost_tcb,prio); */
... Opérations normales ...
/*@ ghost dest_ghost_tcb = dest_tcb->make_ghost(); */
/*@ assert dest_ghost_tcb == res_ghost_tcb; */
```



## Exemple (suite)

### Comportement de sys\_schedule

```
/*@ ghost ghost_tcb& dest_ghost_tcb =
        dest_tcb->make_ghost(); */
/*@ ghost ghost_tcb& current_ghost_tcb =
        current_tcb.make_ghost(); */
/*@ assert dest_ghost_tcb.scheduler ==
        &current_ghost_tcb; */
/*@ assert prio >= 0 && prio <= MAX_PRIO; */
/*@ assert current_ghost_tcb.priority > prio; */
/*@ ghost ghost_tcb& res_ghost_tcb =
        ghost_schedule_prio(dest_ghost_tcb,prio); */
... Opérations normales ...
/*@ ghost dest_ghost_tcb = dest_tcb->make_ghost(); */
/*@ assert dest_ghost_tcb == res_ghost_tcb; */
```



# Avantages et faiblesses



## Avantages

- Se prête plutôt bien à l'analyse de valeur
- Spécification exécutable, prototypage

## Inconvénients

- Code ghost réparti en de multiples endroits
- Vérification du code ghost lui-même

## En résumé

- Analyse de code “réel” possible avec Frama-C
- ACSL utilisable comme langage de modélisation



## Perspectives possibles

- Améliorer le traitement des annotations
- Développement des greffons de WP
- Passer à l'échelle en matière de spécification
- Lien avec des langages de modélisation existants (B, Focal, UML)

