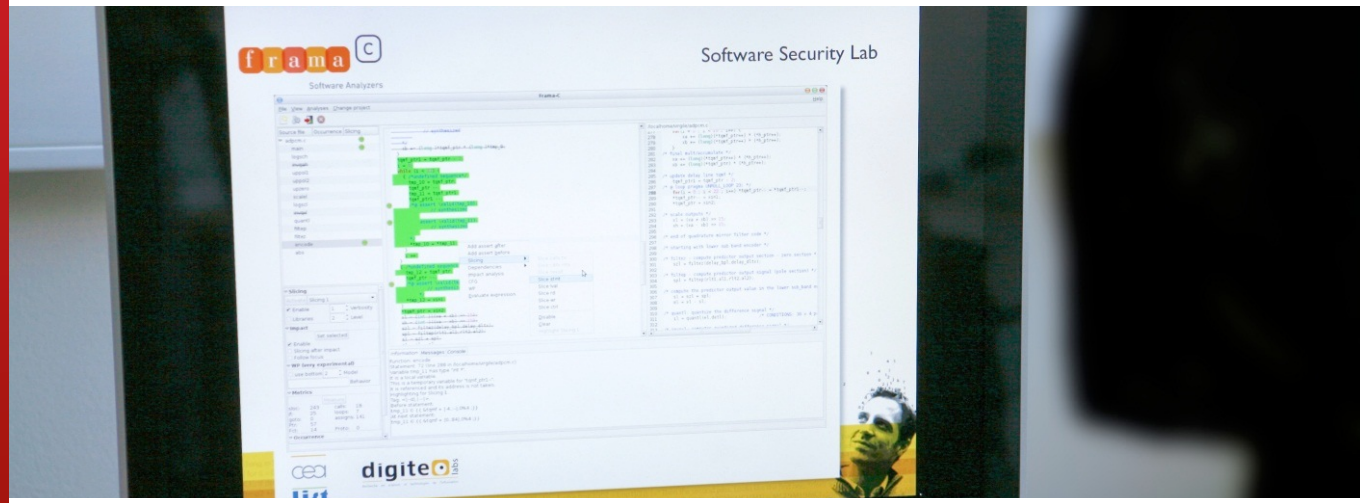
DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

REGION MODEL

```
> frama-c -wp-model region
```



REGION ANALYSIS : FRAMA-C / WP-REGION

Recap :

- ✓ need for **smarter** memory models
- ✓ there exists typical separation **patterns**
- ✓ users shall be able to **specify** separation
- ✓ common patterns shall be **inferred**

Implemented in 2017 :

- ✓ region and pattern annotations
- ✓ memory points-to & data layout analysis
- ✓ inférence of never-alias patterns
- ✓ memory model selection is **almost** there

REGION ANALYSIS : FRAMA-C / WP-REGION

Recap :

- ✓ need for **smarter** memory models
- ✓ there exists typical separation **patterns**
- ✓ users shall be able to **specify** separation
- ✓ common patterns shall be **inferred**

2018 Objectives

- ✓ compound-copies (tracked, but not used)
- ✓ points-from analysis
- ✓ pessimistic *vs.* optimistic aliasing patterns
- ✓ memory model for WP

REGION ANALYSIS : FRAMA-C / WP-REGION

Recap :

- ✓ need for **smarter** memory models
- ✓ there exists typical separation **patterns**
- ✓ users shall be able to **specify** separation
- ✓ common patterns shall be **inferred**

Implemented in 2018

- ✓ compound-copies (tracked, but not used)
- ✓ points-from analysis
- ✓ ~~pessimistic~~ *vs.* optimistic aliasing patterns
- ✓ memory model for WP

DE LA RECHERCHE À L'INDUSTRIE

cea

IMPLEMENTATION

REGION ANNOTATIONS

ACSL Extensions

New Clause for Behaviors

```
@region region-spec, ... ;
```

Region Specifications: represent a collections of memory locations (*aka* « regions »)

```
region-spec ::=
    | ident: region-spec     // The region is given a name
    | \pattern{ident}, ...   // The region shall follow the pattern(s)
    | l-path, ...            // The region contains the given locations
```

Locations: represent sets of l-values (*aka* « l-paths »)

```
l-path ::=
    | ident                 // Named memory region
    | ident                 // C-variables (in scope)
    | ( c-type ) l-path     // C-Casts
    | * l-path              // Pointer access
    | l-path  [ range ]     // Array or Pointer access
    | l-path +( range )     // Pointer shift
    | l-path .  ident       // Field access
    | l-path -> ident       // Field arrow
    | ( l-path.ident? .. l-path.ident? ) // Field range
    | ( l-path->ident? .. l-path->ident? ) // Field arrow range

<range> ::= <exp> | <exp>?..<exp>?
```

REGION ANALYSIS : FRAMA-C / WP-REGION

Extension of Frama-C / WP

Frama-C / WP : New Options

```
-region-annot          Register '@region' ACSL Annotations (auto with
                       -wp-region) (opposite option is -no-region-annot)
-wp-region             Perform Region Analysis (experimental) (opposite option
                       is -wp-no-region)
-wp-region-cluster     Compute region clustering fixpoint (set by default,
                       opposite option is -wp-no-region-cluster)
-wp-region-fixpoint     Compute region aliasing fixpoint (set by default,
                       opposite option is -wp-no-region-fixpoint)
-wp-region-flat        Flatten arrays by default (opposite option is
                       -wp-no-region-flat)
-wp-region-inline      Inline aliased sub-clusters (set by default, opposite
                       option is -wp-no-region-inline)
-wp-region-pack        Pack clusters by default (set by default, opposite option
                       is -wp-no-region-pack)
-wp-region-rw          Written region are considered read-write by default (set
                       by default, opposite option is -wp-no-region-rw)
```

REGION ANALYSIS : FRAMA-C / WP-REGION

Analyses Complémentaires de Region

Frama-C / WP : Visualisation des Analyses

```
-wp-msg-key dot,pdf   Output memory graphs with names <f>.{dot,pdf}
-wp-msg-key offset    Dump l-path access in memory graphs
-wp-msg-key deref     Dump typed access to each region
-wp-msg-key cluster   Dump data-layout analysis in memory graphs
-wp-msg-key from      Dump from analysis
-wp-msg-key roots     Dump roots analysis
-wp-msg-key chunk     Dump memory chunk allocation
-wp-msg-key garbled   Warns on garbled-mix regions
```

REGION ANALYSIS : FRAMA-C / WP-REGION

Infrastructure de Tests pour les régions

Fichiers de tests unitaire (x19)

```
$ cd src/plugins/wp/tests/wp_region
$ ls *.i
annot.i        array4.i       array8.i       index.i        structarray3.i
array1.i       array5.i       fb_ADD.i       matrix.i       structarray4.i
array2.i       array6.i       fb_SORT.i      structarray1.i swap.i
array3.i       array7.i       garbled.i      structarray2.i
```

Script utilisateur (en plus de Frama-C ./bin/ptests.opt)

```
$ cd src/plugins/wp/tests/wp_region
$ ./fc.sh -h
fc.sh [options...] <test.[ic]>
  -h,--help     help and exit
  -D,--delete   clean output directory and exit
  -g,--gui      run in Frama-C Gui
  -r,--region   visualize region graph
  -u,--update   commit region graph in oracle
  -t,--test     run ptests.opt on test file (or all files)
  -q,--qualif   run ptests.opt with test-config qualif
  --open <cmd>  opens pdf with '<cmd>'
  -k <keys>     set message keys
  *             any other Frama-C options
```

DE LA RECHERCHE À L'INDUSTRIE

cea

NOUVELLES ANALYSES

REGION ANALYSIS : FRAMA-C / WP-REGION

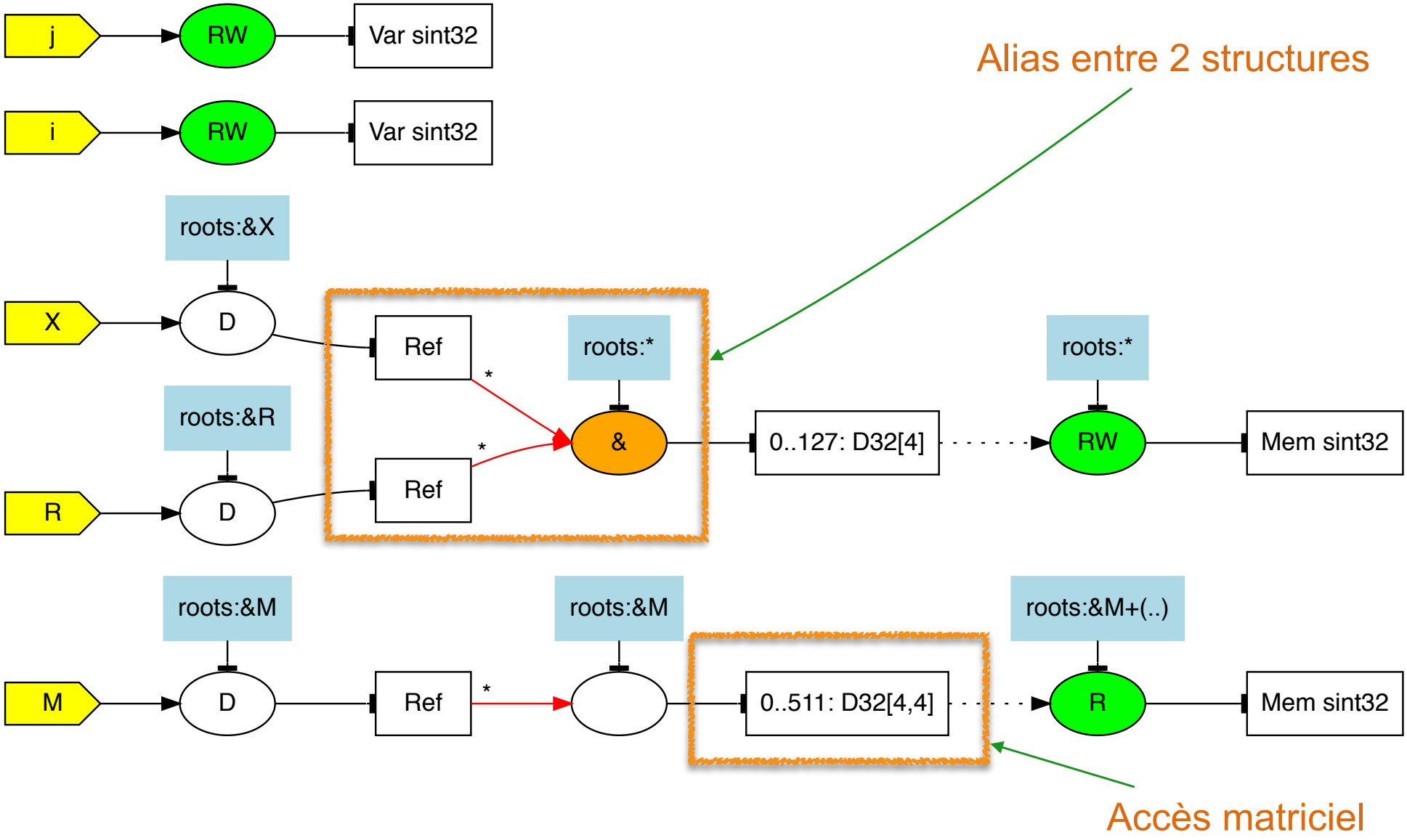
Analyses Matricielles (test struct-array #1)

```
$ cd src/plugins/wp/tests/wp_region
$ ./fb.sh -r structarray1.i

typedef struct Vector {
  int coord[4];
} * vector ;

typedef struct Matrix {
  int coef[4][4];
} * matrix ;

//@ region *X , *R ;

void job( matrix M , vector X , vector R )
{
  for (int i = 0; i < 4; i++) {
    R->coord[i] = 0 ;
    for (int j = 0; j < 4; j++) {
      R->coord[i] += M->coef[i][j] * X->coord[j];
    }
  }
}
```

Alias entre 2 structures

Accès matriciel

REGION ANALYSIS : FRAMA-C / WP-REGION



REGION ANALYSIS : FRAMA-C / WP-REGION

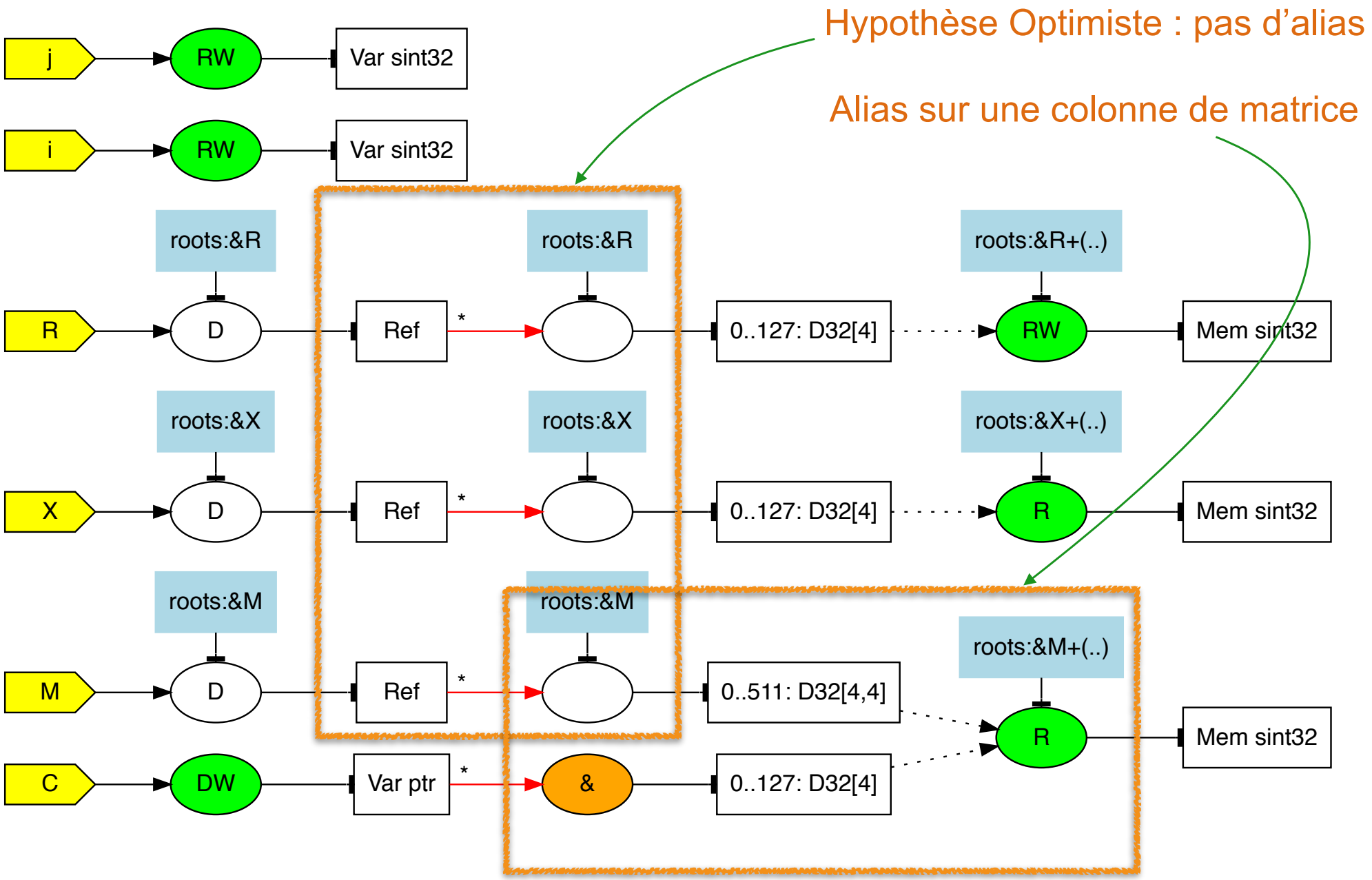
Analyses Matricielles (test struct-array #2)

```
$ cd src/plugins/wp/tests/wp_region
$ ./fb.sh -r structarray2.i

typedef struct Vector {
  int coord[4];
} * vector ;

typedef struct Matrix {
  int coef[4][4];
} * matrix ;

void job( matrix M , vector X , vector R )
{
  for (int i = 0; i < 4; i++) {
    R->coord[i] = 0 ;
    for (int j = 0; j < 4; j++) {
      vector C = (vector) (M->coef[i]) ;
      R->coord[i] += C->coord[j] * X->coord[j];
    }
  }
}
```

Hypothèse Optimiste : pas d'alias

Alias sur une colonne de matrice

REGION ANALYSIS : FRAMA-C / WP-REGION



REGION ANALYSIS : FRAMA-C / WP-REGION

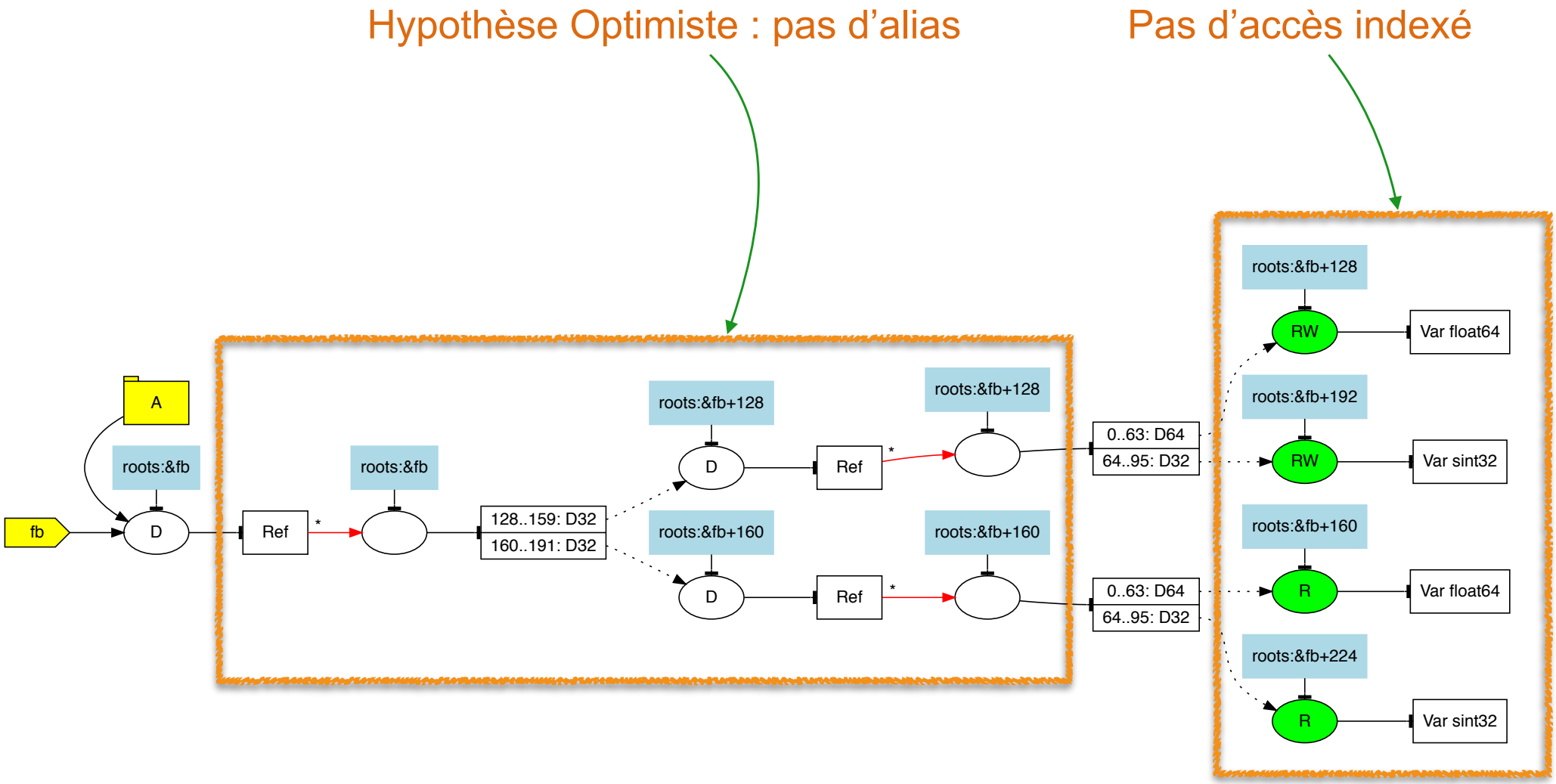
Analyses de Root (test fb_ADD)

```
$ cd src/plugins/wp/tests/wp_region
$ ./fb.sh -r fb_ADD.i

typedef struct N { double v ; int s ; } *SN ;
typedef struct L { int v ; int s ; } *SL ;

typedef struct Block {
  SN prm ;
  SN inp1 ;
  SN inp2 ;
  SN inp3 ;
  SN out1 ;
  SN out2 ;
  SN out3 ;
  SL idx1 ;
  SL idx2 ;
  SL idx3 ;
  SN sum ;
} FB ;

/*@
  region A: fb ;
*/
void job(FB *fb)
{
  fb->out1->v = fb->out1->v + fb->out2->v ;
  fb->out1->s = fb->out1->s | fb->out2->s ;
}
```

Hypothèse Optimiste : pas d'alias

Pas d'accès indexé

REGION ANALYSIS : FRAMA-C / WP-REGION



REGION ANALYSIS : FRAMA-C / WP-REGION

Analyses de Root (test fb_SORT)

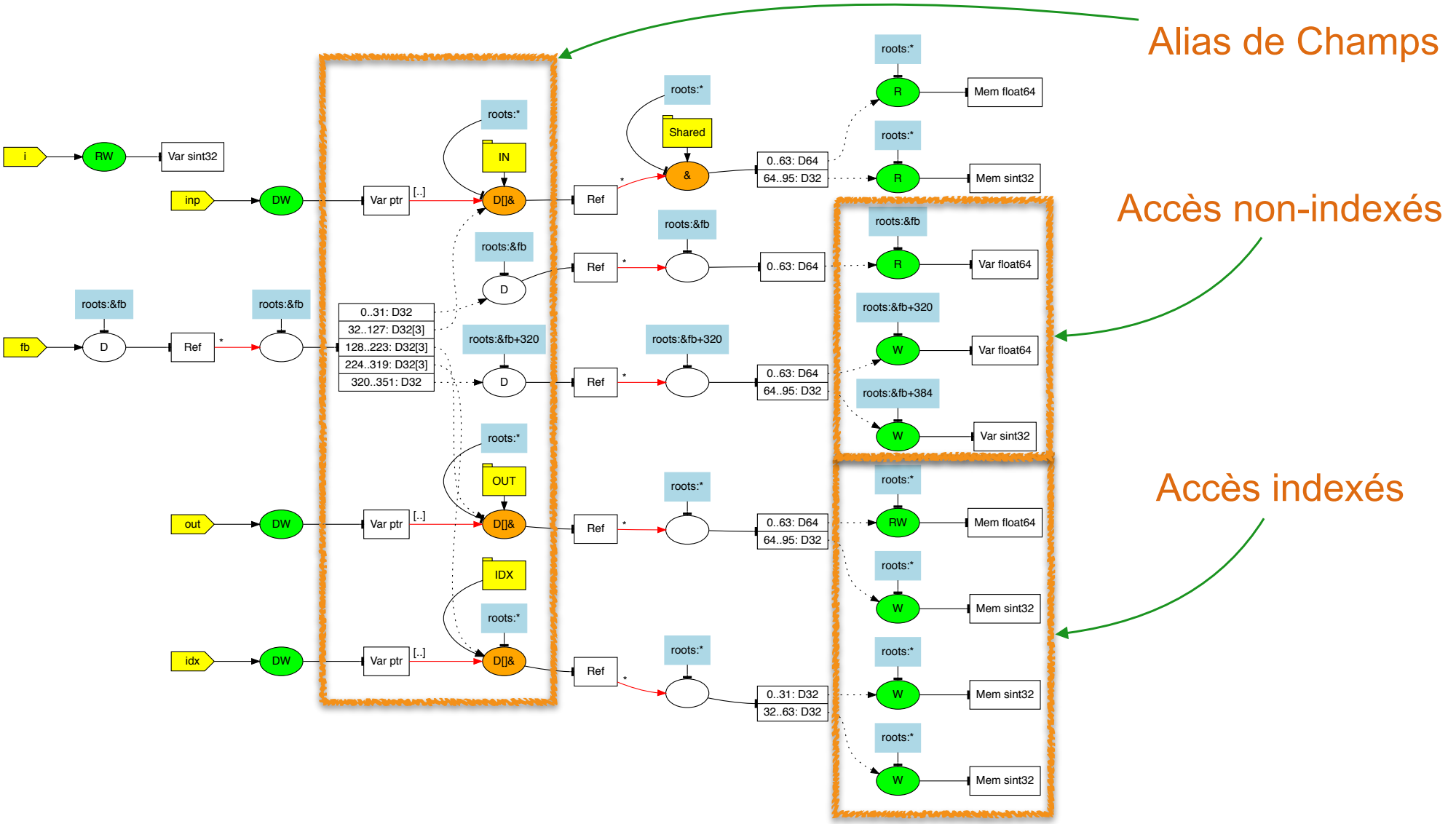
```
$ cd src/plugins/wp/tests/wp_region
$ ./fb.sh -r fb_SORT.i

/*@
  region Shared: *(fb->inp1 .. fb->inp3);
  region IN:      (fb->inp1 .. fb->inp3);
  region OUT:     (fb->out1 .. fb->out3);
  region IDX:     (fb->idx1 .. fb->idx3);
 */
void job(FB *fb)
{
  SN *inp = &(fb->inp1) ;
  SN *out = &(fb->out1) ;
  SL *idx = &(fb->idx1) ;

  for (int i = 0; i < 3; i++) {
    out[i]->v = inp[i]->v + fb->prm->v ;
    out[i]->s = 0 ;
    idx[i]->v = inp[i]->s ;
    idx[i]->s = 0 ;
  }

  fb->sum->v =
    fb->out1->v +
    fb->out2->v +
    fb->out3->v ;

  fb->sum->s = 0 ;

}
```

Alias de Champs

Accès indexés

Accès non-indexés

REGION ANALYSIS : FRAMA-C / WP-REGION



DE LA RECHERCHE À L'INDUSTRIE

cea

MODELE MEMOIRE

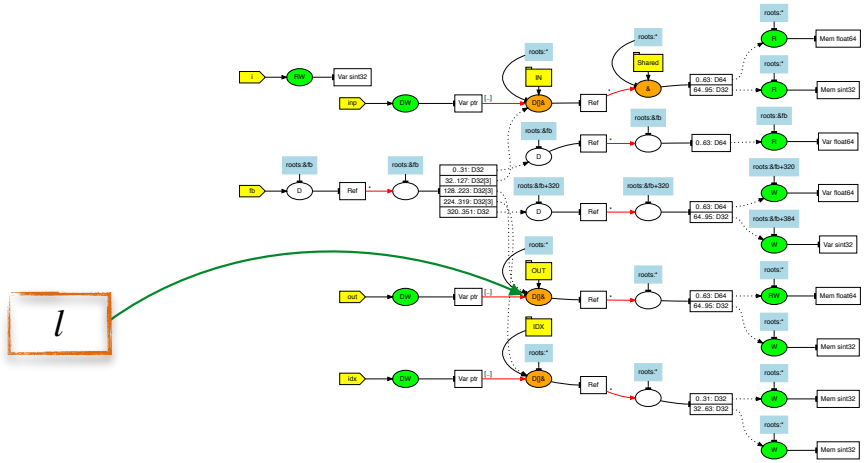
REGION MODEL

Modèle CompCert (de référence)

Adresse : $A ::= \{\ \text{addr} : \mathbb{Z}\ ;\ \text{offset} : \mathbb{Z}\ \}$

Mémoire : $\text{read\_sint32} : (M, A) \to \mathbb{Z}$

$\text{write\_sint32} : (M, A, \mathbb{Z}) \to M'$



Modèle Region

Index : $l \in L \qquad r \in R$ (position dans le graphe des régions)

Concrétisation : $\text{addrof} : L \to A$ (adresse d'un index, non-injectif)

Cohérence : $\text{consistent} : L \to \text{Prop}$ (respect des plages d'index autorisées)

Mémoire (non-indexée) :

$x_r : \mathbb{Z}$ (région (*r*) représentant *un* unique entier)

$x_r : L$ (région (*r*) représentant *un* unique pointeur)

(pour les pointeurs, on mémorise **l'index** et non l'adresse pour conserver le chemin d'origine dans le graphe)

Mémoire (indexée) :

$m_r : [A \mapsto \mathbb{Z}]$ (région (*r*) représentant *des* entiers)

$m_r : [A \mapsto L]$ (région (*r*) représentant *des* pointeurs)

REGION MODEL (CORRECTION)

Modèle CompCert (de référence)

Adresse : $A ::= \{\ \text{addr} : \mathbb{Z}\ ;\ \text{offset} : \mathbb{Z}\ \}$

Mémoire : $\text{read\_sint32} : (M, A) \to \mathbb{Z}$

$\text{read\_pointer} : (M, A) \to A$

Modèle Region

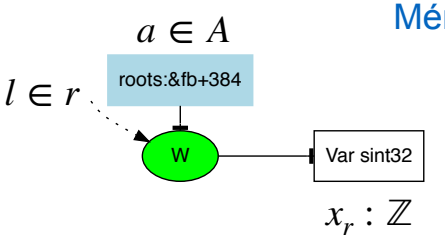
Index : $l \in L \qquad r \in R$

Concrétisation : $\text{addrof} : L \to A$

Cohérence : $\text{consistent} : L \to \text{Prop}$

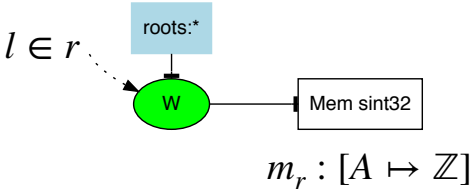
Propriétés de Cohérence :

Mémoire (entier non-indexé) :



$$\forall l \in r, \text{consistent}(l) \implies \text{addrof}(l) = a \ \wedge\ x_r = \text{read\_sint32}(M, a)$$

Mémoire (entiers indexés) :



$$\forall l \in r, \text{consistent}(l) \implies m_r[\text{addrof}(l)] = \text{read\_sint32}(M, \text{addrof}(l))$$

REGION MODEL (CORRECTION)

Modèle CompCert (de référence)

Adresse : $A ::= \{\ \text{addr} : \mathbb{Z}\ ;\ \text{offset} : \mathbb{Z}\ \}$

Mémoire : $\text{read\_sint32} : (M, A) \to \mathbb{Z}$

$\text{read\_pointer} : (M, A) \to A$

Modèle Region

Index : $l \in L \qquad r \in R$

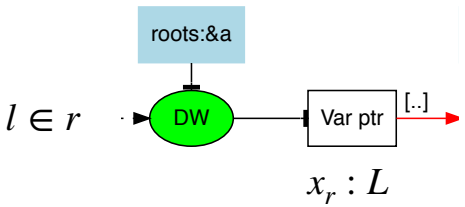
Concrétisation : $\text{addrof} : L \to A$

Cohérence : $\text{consistent} : L \to \text{Prop}$

Propriétés de Cohérence :

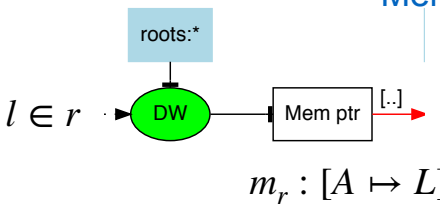
$a \in A$

Mémoire (pointeur non-indexé) :



$\forall l \in r, \text{consistent}(l) \implies \text{addrof}(l) = a \ \wedge\ \text{addrof}(x_r) = \text{read\_pointer}(M, a)$

Mémoire (pointeurs indexés) :



$\forall l \in r, \text{consistent}(l) \implies \text{addrof}(m_r[\text{addrof}(l)]) = \text{read\_pointer}(M, \text{addrof}(l))$

REGION MODEL (INDEX)

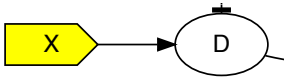
Index : $l \in L \qquad r \in R$

Concrétisation : addrof : $L \to A$

Cohérence : consistent : $L \to$ Prop

Modèles & Cohérence des Index

Variables :



$l = \text{Global}_X$

$$\text{addrof}(\text{Global}_X) = \{\text{base} = \text{Base}_X\,;\ \text{offset} = 0\}$$

$$\text{consistent}(\text{Global}_X) \iff \text{true}$$

REGION MODEL (INDEX)

Index : $l \in L \qquad r \in R$

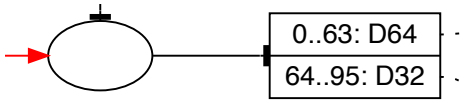
Concrétisation : addrof : $L \rightarrow A$

Cohérence : consistent : $L \rightarrow$ Prop

$\{ \text{ base} = a \text{ ; offset} = i \} \oplus k = \{ \text{ base} = a \text{ ; offset} = i + k \}$

Modèles & Cohérence des Index

Structures : $l \in r$



$l_1 = \text{Field}_{\{0,64\}}(l_0, 0)$

$l_2 = \text{Field}_{\{0,64\}}(l_0, 64)$

$$\text{addrof}(\text{Field}_S(l,k)) = \text{addrof}(l) \oplus k$$

$$\text{consistent}(\text{Field}_S)(l,k) \iff \text{consistent}(l) \ \wedge \ k \in S$$

REGION MODEL (INDEX)

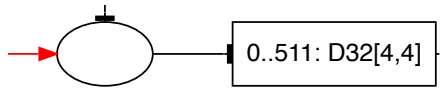
Index : $l \in L \qquad r \in R$

Concrétisation : addrof : $L \rightarrow A$

Cohérence : consistent : $L \rightarrow$ Prop

Modèles & Cohérence des Index

$\{ \text{ base} = a \text{ ; offset} = i \} \oplus k = \{ \text{ base} = a \text{ ; offset} = i + k \}$

Tableaux : $l \in r$ 

$l' = \text{Array}_{32,[4,4]}(l, i, j)$

$$\text{addrof}(\text{Array}_{s,ds}(l, k_1 \ldots k_n)) = \text{addrof}(l) \oplus s \,.\, \Sigma_i(\, k_i \,.\, \Pi_{i<j} d_j \,)$$

$$\text{consistent}(\text{Array}_{s,ds}(l, k_1 \ldots k_n)) \iff \text{consistent}(l) \;\wedge\; \forall i,\, 0 \leq k_i < d_i$$

REGION MODEL (INDEX)

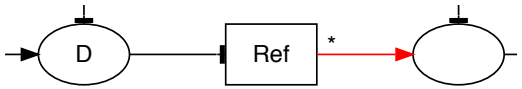
Index : $l \in L \qquad r \in R$

Concrétisation : addrof : $L \rightarrow A$

Cohérence : consistent : $L \rightarrow$ Prop

Modèles & Cohérence des Index

Références : $l \in r$



$l' = \text{Ref}(l)$

$\text{addrof}(\text{Ref}(l)) = \varphi(\text{addrof}(l))$ $\qquad \varphi : A \rightarrow A$ (injective)

$\text{consistent}(\text{Ref}(l)) \iff \text{consistent}(l)$

DE LA RECHERCHE À L'INDUSTRIE

cea

CONCLUSION

REGION ANALYSIS : FRAMA-C / WP-REGION

Implemented in 2018

- ✓ compound-copies (tracked, but not used)
- ✓ points-from analysis
- ✓ ~~pessimistic~~ *vs.* optimistic aliasing patterns
- ✓ memory model for WP
- ✓ consistency checks generated, but not yet verified
- ★ unsupported ACSL fonctions & predicates

2019 Objectives

- ✓ compound-copies (tracked, but not used)
- ✓ memory patterns
- ✓ consistency checks verification
- ✓ full ACSL support

DE LA RECHERCHE À L'INDUSTRIE

cea

THANKS !