PolySpace TECHNOLOGIES

Vérification statique de programmes par interprétation abstraite: principes et expériences industrielles

Daniel Pilaud

Vérification logicielle
Objectifs généraux

- Respect des spécifications fonctionnelles par le programme
  - Par simulation, tests, preuves formelles, dérivation automatique…
- Respect des contraintes temporelles
  - Temps de réponse, respect des cadences d'échantillonnage, terminaison
- Absence d'erreurs d'exécution et de non-déterminisme

Description générale du processus de vérification

- *Extraction automatique de propriétés approchées du programme par interprétation abstraite*
- *Vérification des conditions de correction du programme à l ’aide des propriétés approchées*
- *Identification exhaustive des erreurs d’exécution*

Erreurs détectées

- accès concurrents à des variables partagées non protégées
- accès en lecture à des variables non initialisées
- code non atteignable
- accès hors des bornes d'un tableau et débordement de buffers
- débordement arithmétique sur les entiers et les flottants
- exceptions arithmétiques : division par 0, racine carrée d'un nombre négatif …
- conversion illégale de types

Exemple

Y-a-t'il des erreurs dans cette procédure ?

Si oui, combien ?

```
procedure foo
        (RR:in Float64; F1:in Float64;
        F2:in out Float64; F3:in out Float64)is
tmp:Float64;
begin
if (RR /= 0.0) then
        F2 := F1 / RR;
        ...
        tmp := RR**2;
        F3 := F1 / tmp;
end if;
end foo;
```

Exemple

Résultats générés :

- *Type d ’erreurs*
- *Localisation précise dans le code source*
- *Description du contexte d ’erreur*

```
* possible failure of correctness condition: denominator
must be non zero
computed range: {0<=[expr]<=10**10}
in task1.proc_val at "validate.adb" line 24, column 18:
     F3 := F1 / tmp;
               ^
```

Principes techniques

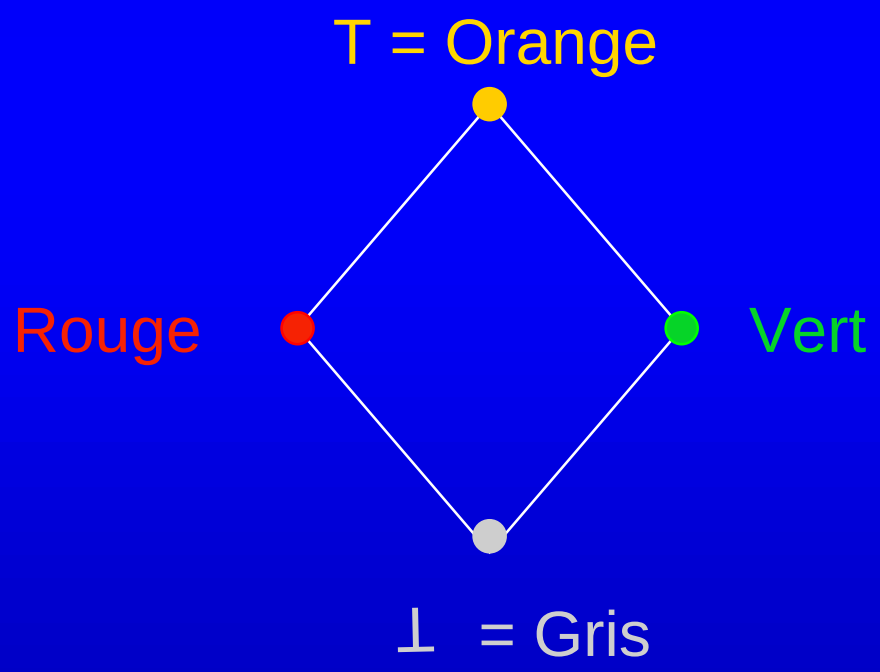
- Analyse du flot de données et de contrôle
  - Construction du graphe d'appel
  - Dictionnaire des variables globales
  - Génération des conditions de correction
  - Accès concurrents à des variables partagées

Principes techniques

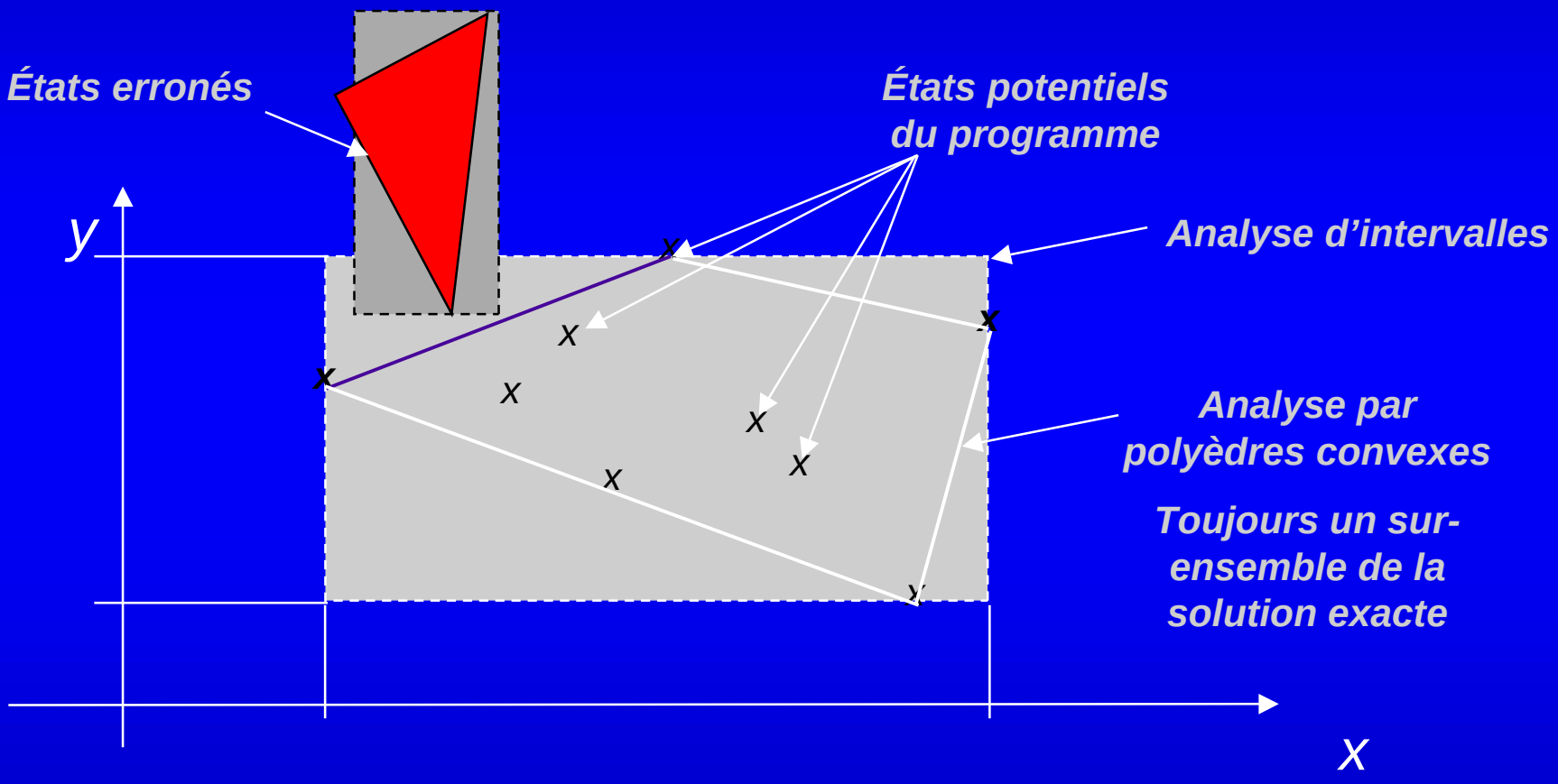
- Analyse de sûreté de fonctionnement
  - Calcul des propriétés approchées des variables
  - Vérification des conditions de corrections à l'aide de ces propriétés
  - Classification des conditions de correction :
    - erreur certaine, erreur potentielle, toujours correct, non exécutable.
  - Une opération correcte peut être classée en erreur potentielle.
  - Une opération causant réellement une erreur est nécessairement classée en erreur potentielle ou certaine.

Sûreté

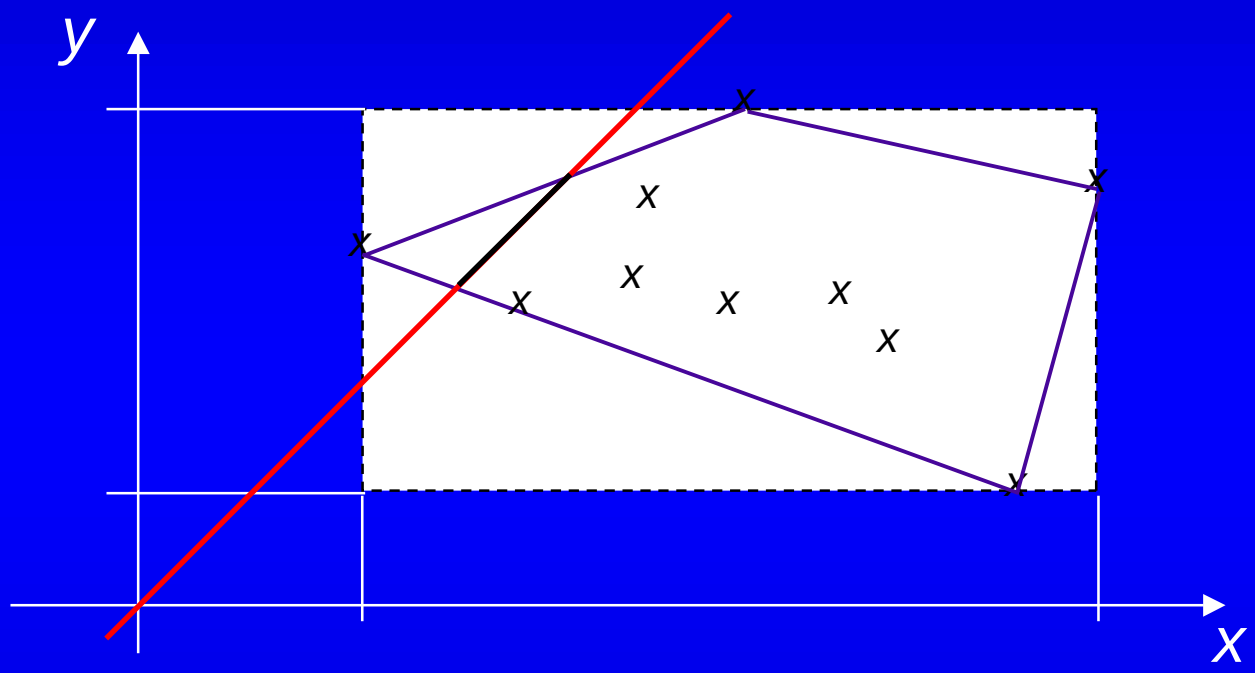
- Une opération causant effectivement une erreur est **nécessairement** classée en erreur potentielle ou certaine.



Principes Techniques



Erreur Potentielle

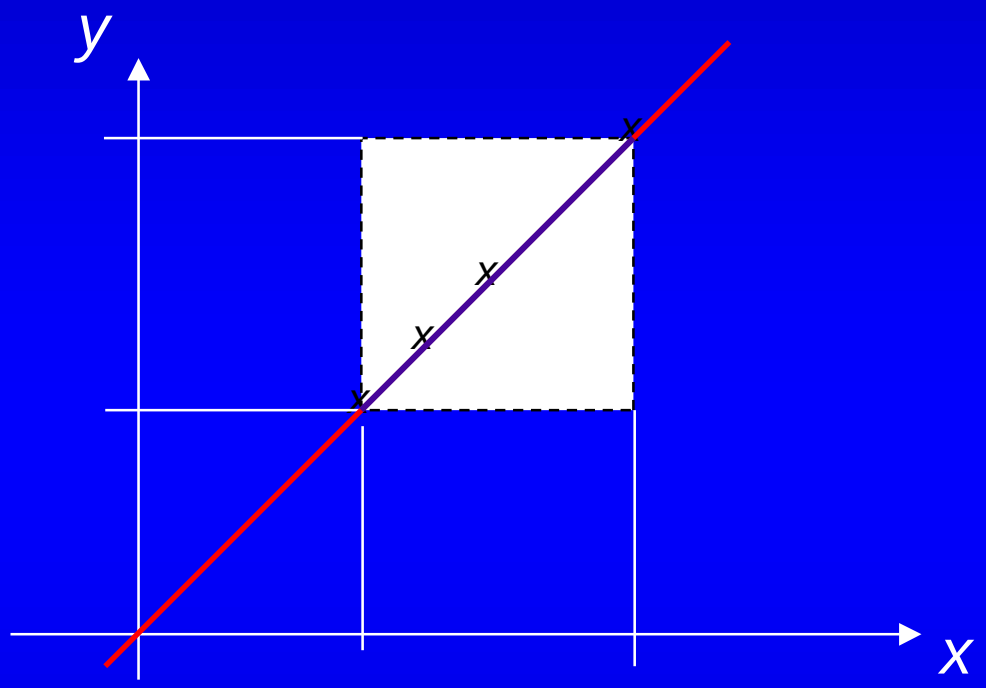


Expression -> a = x / (x-y);

Vérification de la condition x= y

L'intersection entre l 'ensemble des états et celui des états erronés ou celui des conditions d 'erreurs est non vide.

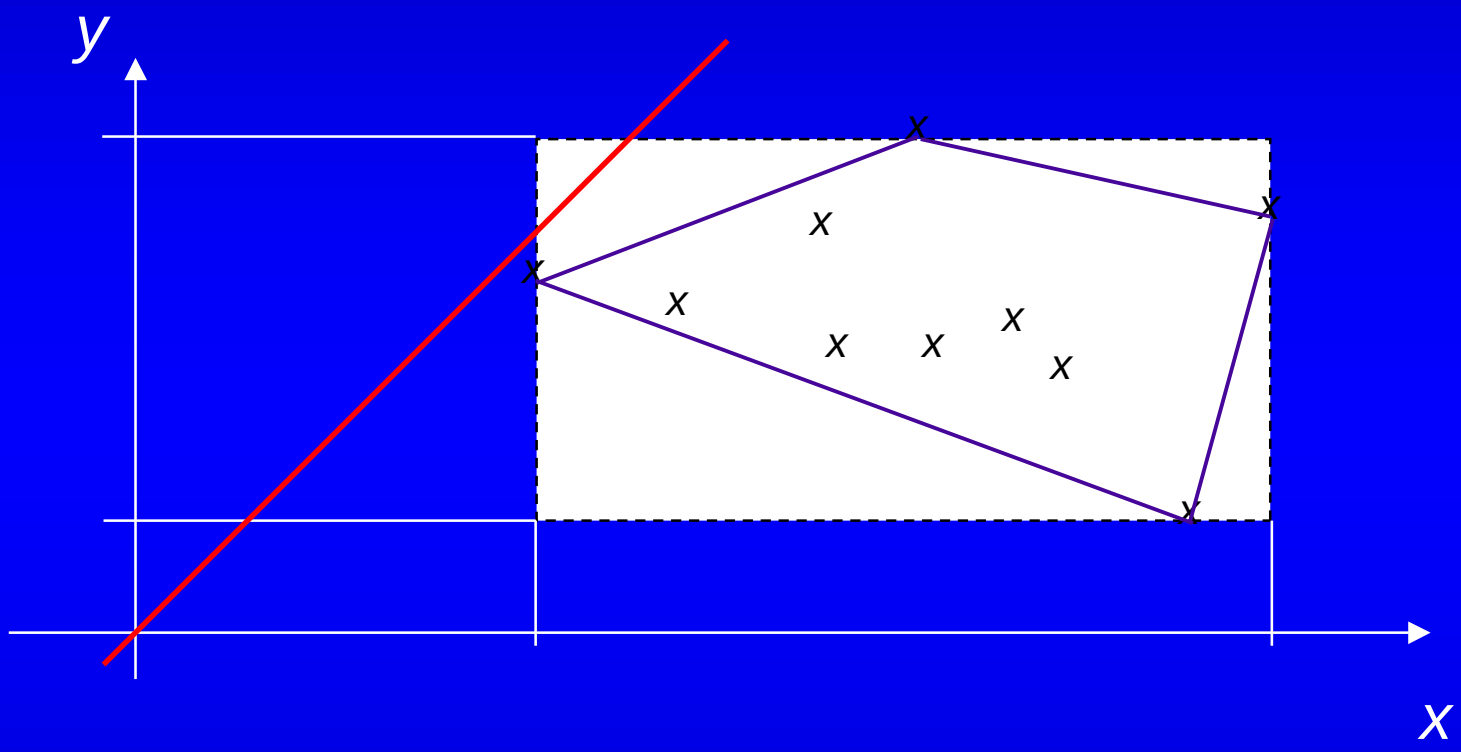
Erreur Certaine



Check : a = x / (x-y);

L'ensemble des états du programme est inclus dans l'ensemble des conditions d'erreur.

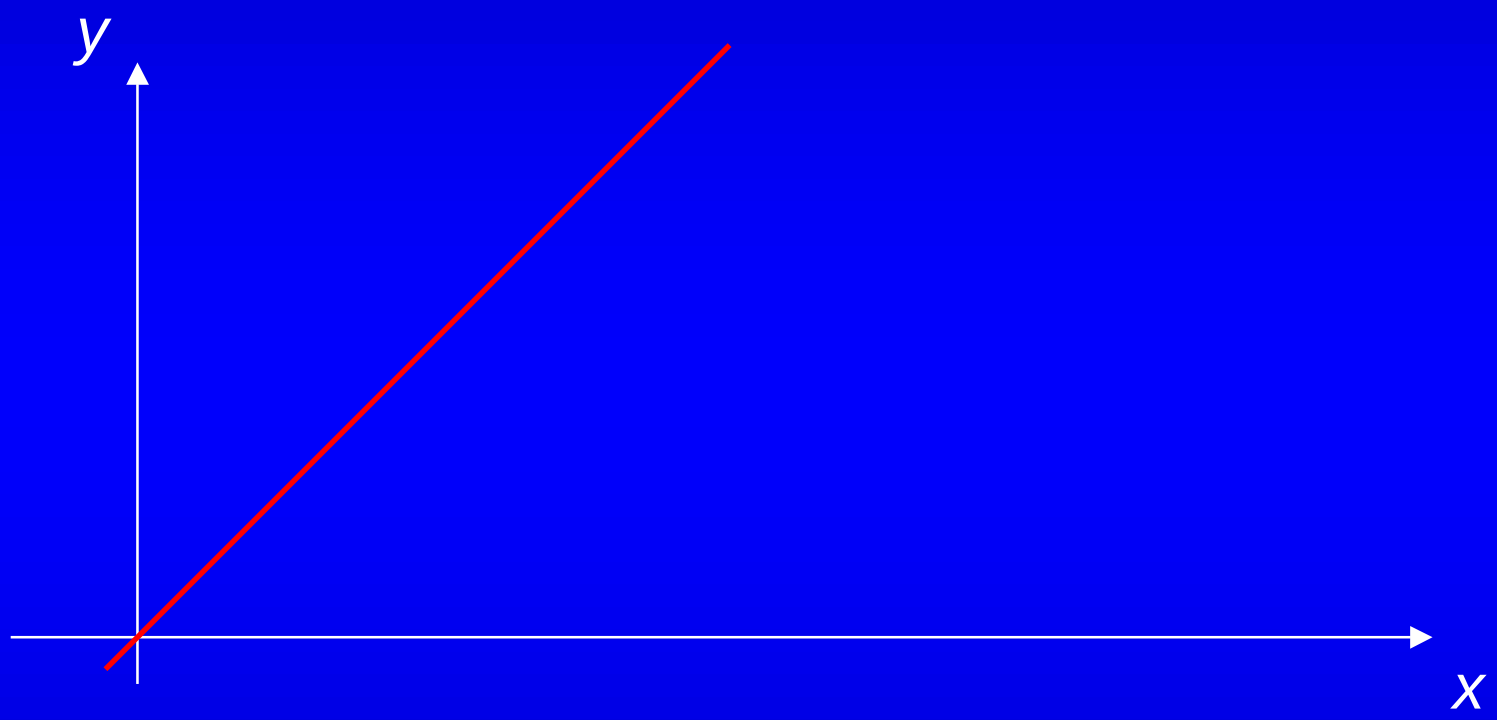
Pas d'erreur



Check : a = x / (x-y);

Intersection vide entre l 'ensemble des états du programme et les conditions d 'erreurs.

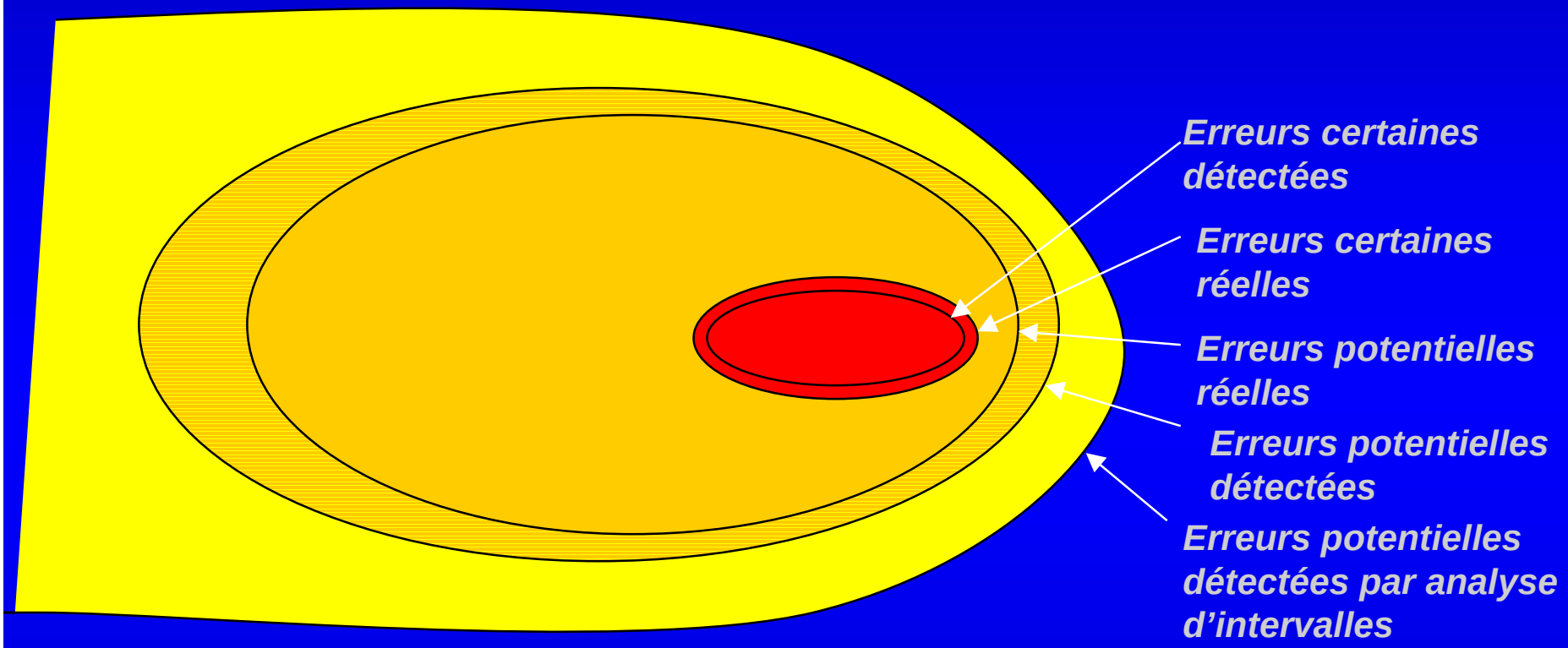
Code non atteignable



Vérification: a = x / (x-y);

L'espace d'états du programme est vide;

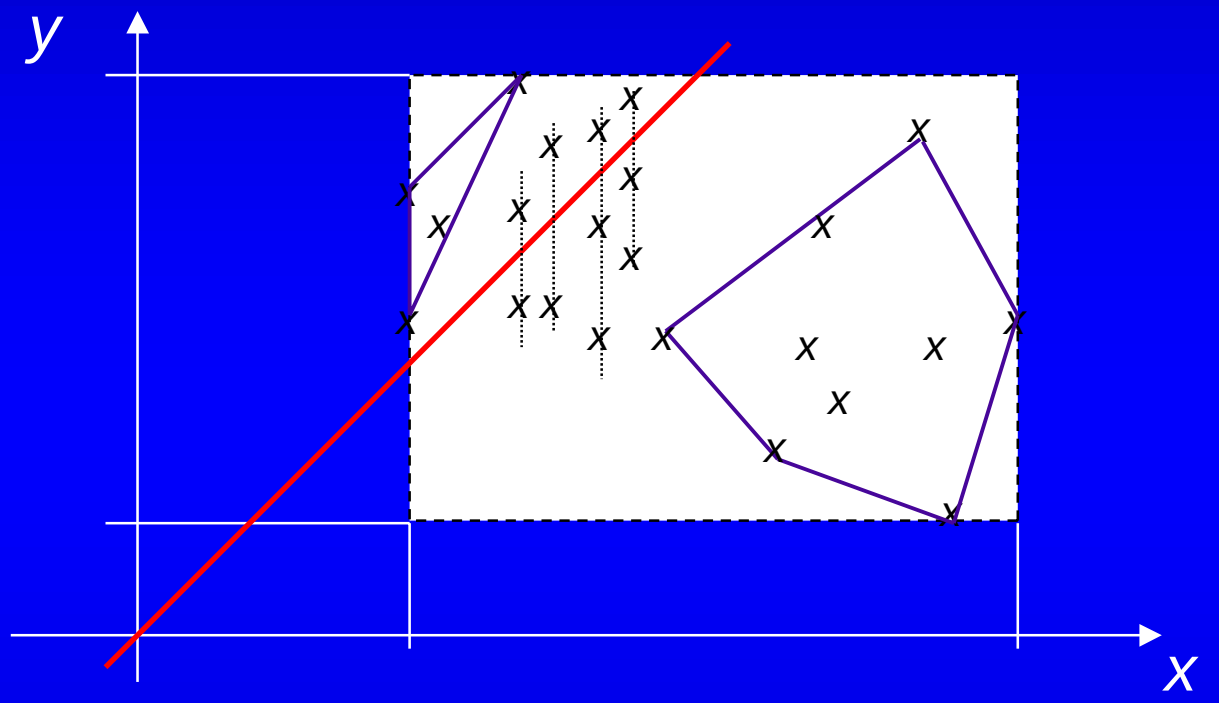
Taux de sélectivité



Erreur certaine : toute exécution de cette partie de code aboutit à une erreur.

Erreur potentielle : il existe une exécution de cette partie de code aboutissant à une erreur.

Optimisations



Plusieurs algorithmes d'optimisation sont nécessaires pour atteindre un bon taux de sélectivité:

- *Construction d'union de polyèdres.*
- *Utilisation de réseaux de points*

Expériences industrielles

- ARIANE: A502, A503
  - Temps réel critique : navigation, guidage...
  - 5 tâches parallèles
  - 60000 LOC

Expérience Ariane 502

- *Analyse flots de données et de contrôle : graphes d'appel, identification des variables globales, détection des variables non initialisées;*
- *Analyse des accès des variables globales : détection des conflits;*
- *Analyse des scalaires : calcul des domaines des variables et vérification des conditions de correction;*
- *Flottants : calcul des domaines des variables et vérification des conditions de correction.*

Expérience Ariane 502

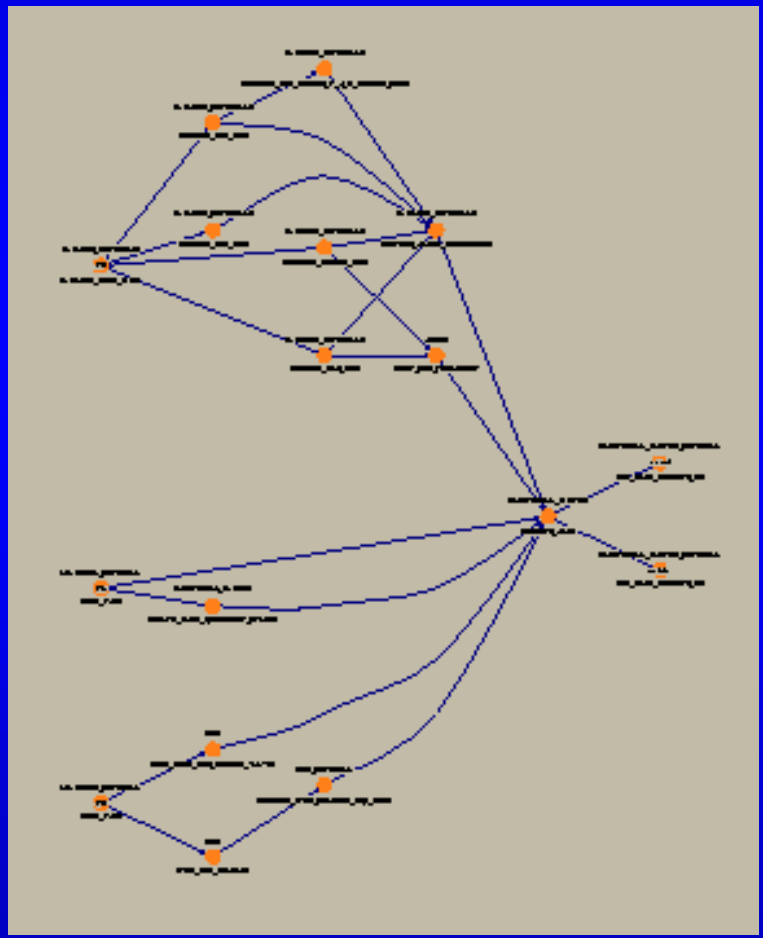
- *Analyse flots de données et de contrôle : graphes d'appel, identification des variables globales, détection des variables non initialisées;*

```
Global variables aliased (A), shared (S), read but not written (R), modified constant (M).

     | variable                    | type                                  | usage pattern
   1 | <ASM-GLOBALS>               |                                       | {S}
   2 | TASK_AUTO.WRONG_AUTO_TESTS  | array(ram..at16) of false..true       | {R}
   3 | TASK_AUTO.RECURSIVE_STATE   | float(32) range -3.41E+38..3.4E+38    | {R}
   4 | TASK_AUTO.N_WAIT1           | 0..255                                | {R}
   5 | TASK_AUTO.N_WAIT2           | 0..255                                | {R}
   6 | TASK_AUTO.N_WAIT3           | 0..255                                | {R}
   7 | TASK_AUTO.N_WAIT4           | 0..255                                | {R}
.......
  13 | AUTOTESTS.TEST_ROM_MEM      | 0..65535                              | {S}
  14 | ENCODING.ARRAY_1_FLAG       | array(0..1023) of 0..255              | {R}
.......
 203 | AR.TB_AR                    | array(pro1..pro2, 1..80) .....2**31-1| {R}
 204 | TEMPO.C_DEC_TEMPO           | 0..2**31-1                            | {R}
 205 | TEMPO.VAL_INIT_TEMPO        | 0..30                                 | {R}
```

Expérience Ariane 502

- *Analyse des accès des variables globales : un graphe par conflit d'accès détecté*



Expérience Ariane 502

- *Analyse des scalaires et des flottants : calcul des domaines des variables et vérification des conditions de correction;*

Module	:	scalars		floats		VNI	
TEST_AUTO	:	40%	6	100%	0	100%	0
CHANNELS_T	:	50%	3	100%	0	100%	0
CODING_C	:	56%	22	100%	0	100%	0
CODING_S	:	26%	57	75%	1	100%	0
CTRL_C	:	100%	0	100%	0	100%	0
DEFCOMMONS	:	100%	0	100%	0	100%	0
FUNCTIONS_TELE	:	50%	8	100%	0	100%	0
G_MANAG_MOTION	:	50%	12	38%	8	100%	0
INITIALIZATION	:	100%	0	100%	0	100%	0
INTERFACE_T	:	75%	3	100%	0	100%	0
...							
...							
INTERFACE_V1	:	64%	72	0%	3	100%	0
INTERFACE_V2	:	65%	8	100%	0	100%	0
LINK_SERIAL	:	100%	0	100%	0	100%	0
LOCATE	:	73%	4	50%	4	100%	0
LOGICS	:	38%	45	100%	0	100%	0
MOD_ENG	:	100%	0	100%	0	100%	0
PRODUCE_AND_DISPLAY	:	56%	7	100%	0	100%	0
PROTOCOL	:	100%	0	100%	0	100%	0
RECEIVE	:	100%	0	100%	0	100%	0
TEMPO	:	81%	5	100%	0	100%	0
total	:	57%	399	22%	45	100%	0

Expériences industrielles

- Atmospheric Reentry Demonstrator
  - Temps réel critique : navigation, guidage...
  - 3 tâches parallèles
  - 26 000 LOC

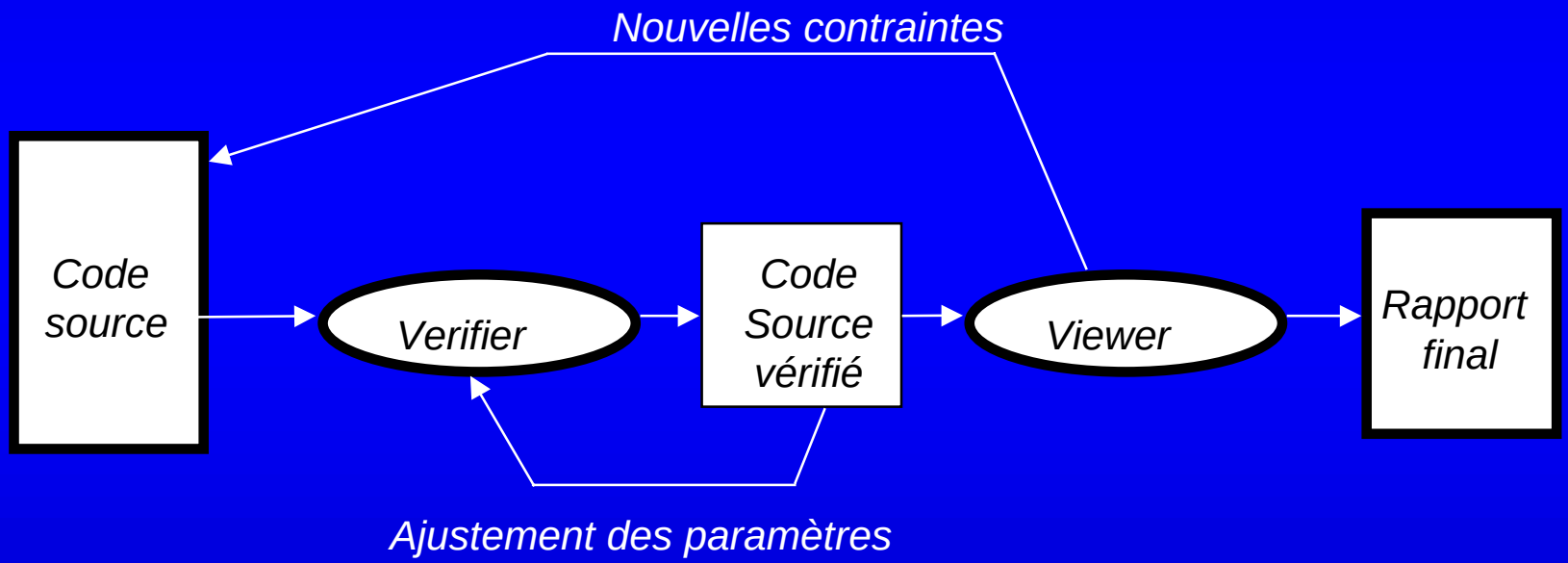
UTILISATIONS

- *Une dizaine d'expériences dans:*
  - *Systèmes embarqués ferroviaires*
  - *avionique*
  - *Energie*
  - *satellites*

Conclusions sur les expériences

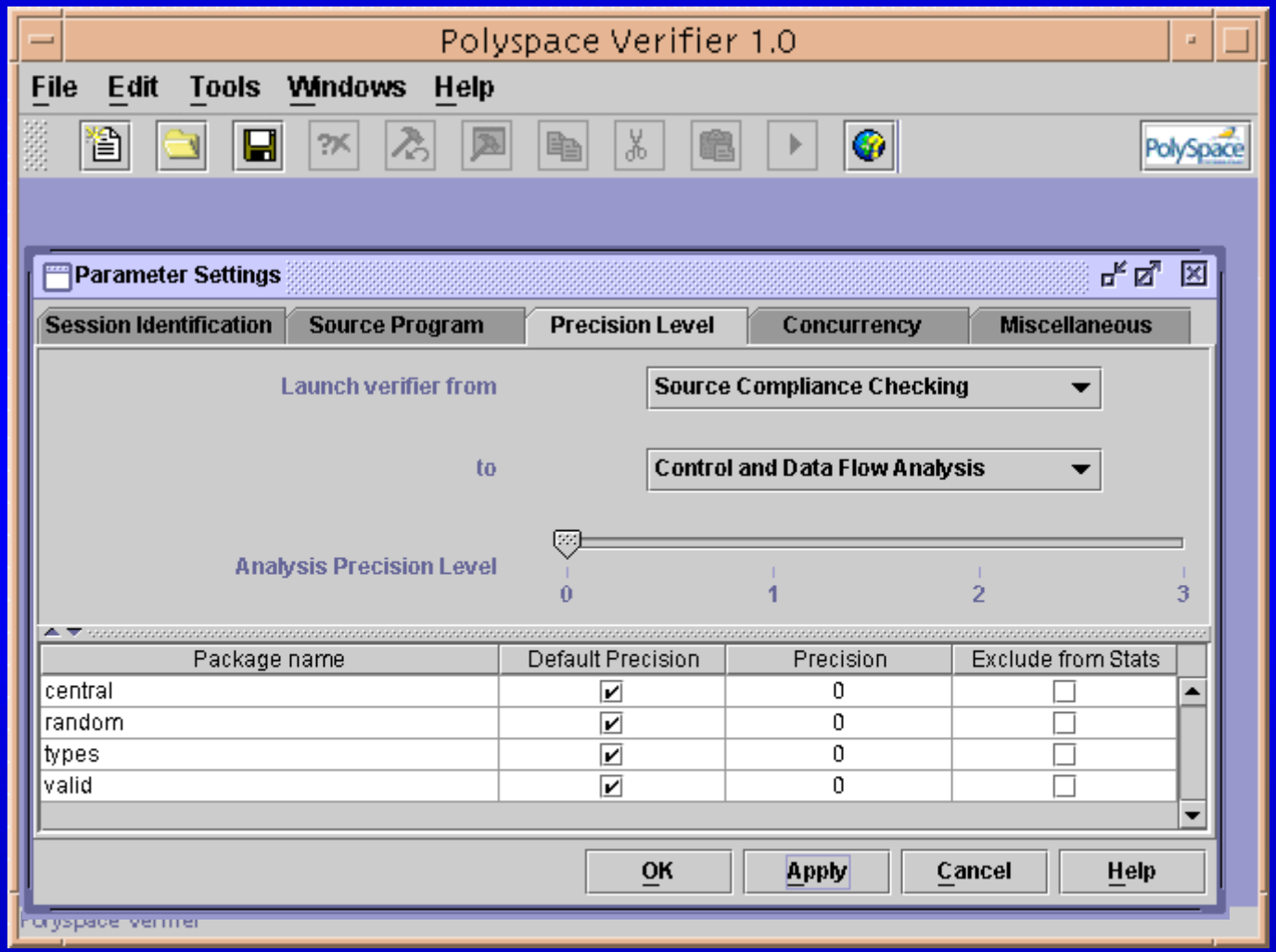
- *Taux de sélectivité élevé (80-95% des opérations à risques)*
- *Revue de code focalisé*
- *Détection plus tôt et exhaustive des erreurs*
  - *sûreté améliorée*
  - *Répétabilité*
  - *coût modéré*

Architecture du produit



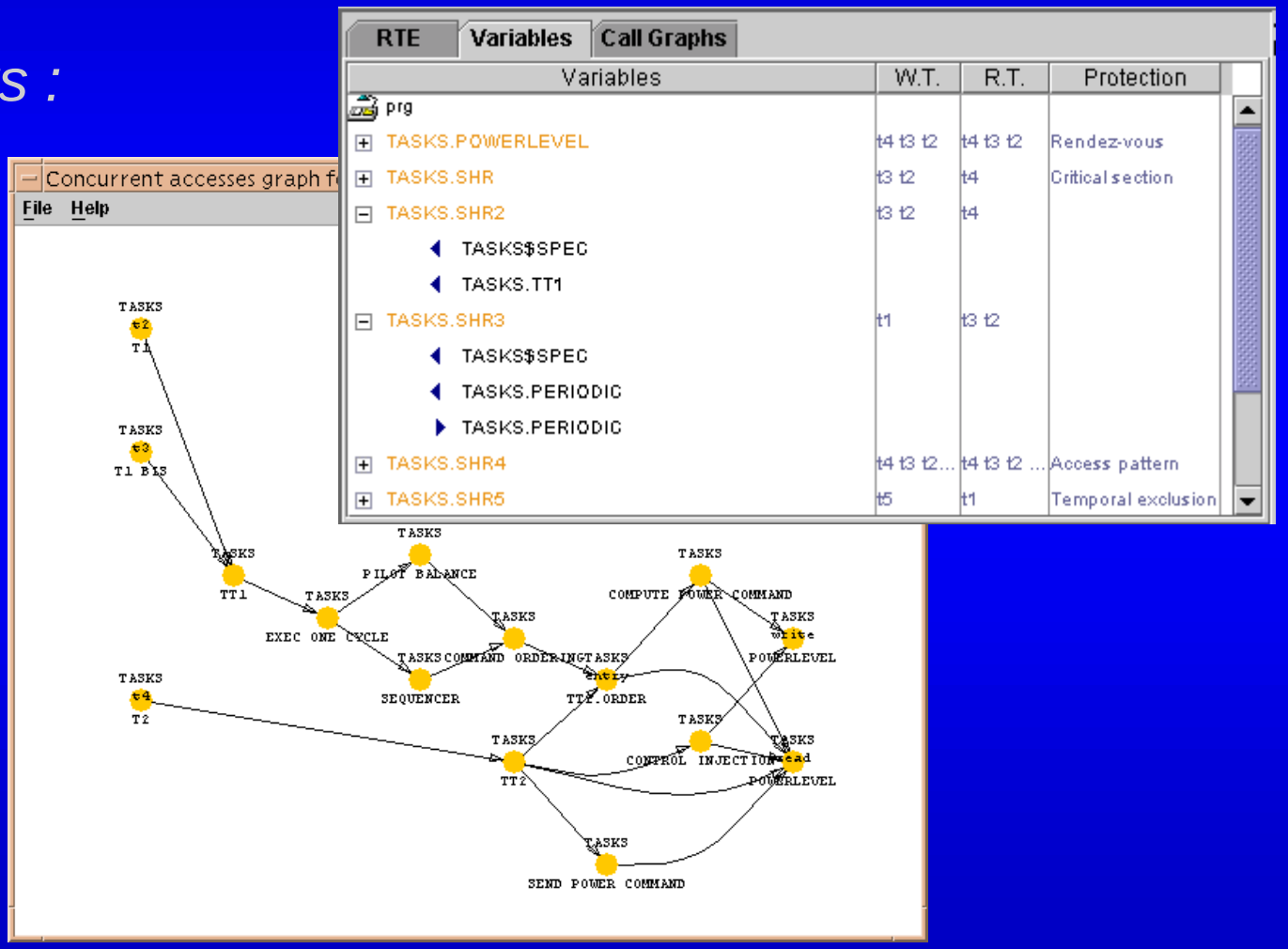
PolySpace Verifier

- *Lancement de la vérification statique*



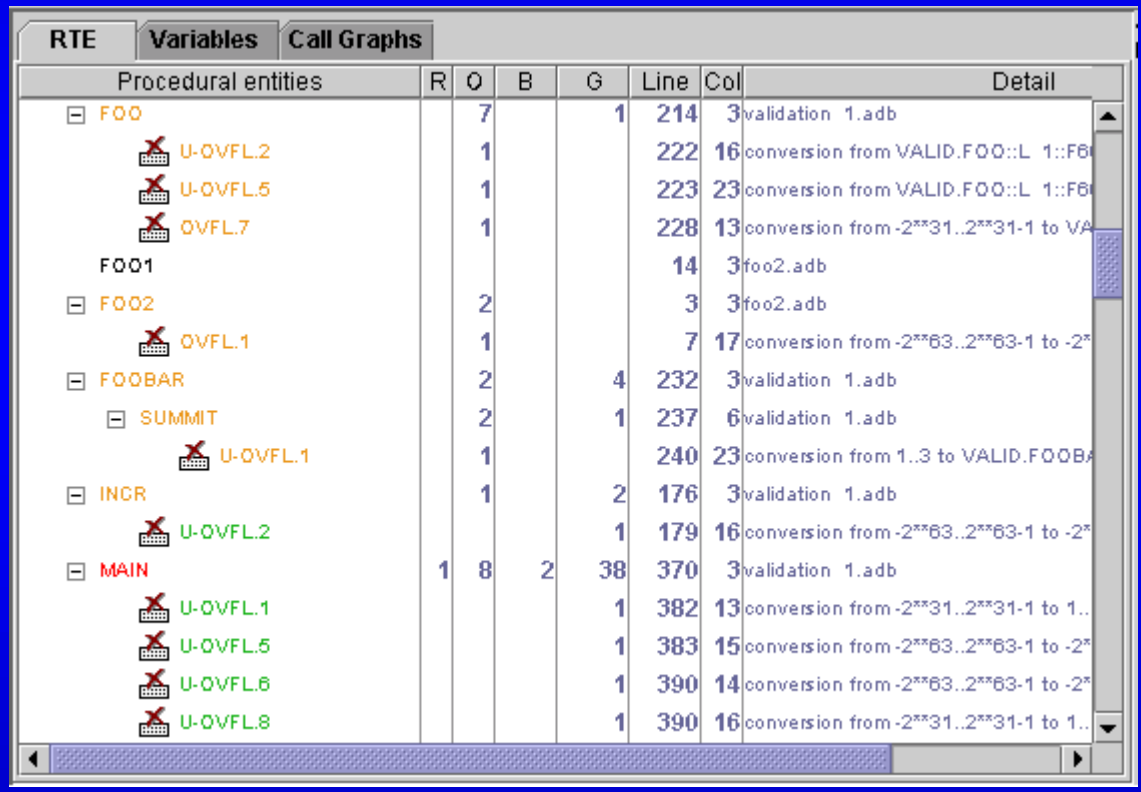
PolySpace Viewer

- *Exploitation des résultats : Variables partagées*



PolySpace Viewer

- *Exploitation des résultats : RTE*



Procedural entities	R	O	B	G	Line	Col	Detail
FOO		7		1	214	3	validation 1.adb
U-OVFL.2		1			222	16	conversion from VALID.FOO::L 1::F6
U-OVFL.5		1			223	23	conversion from VALID.FOO::L 1::F6
OVFL.7		1			228	13	conversion from -2**31..2**31-1 to VA
FOO1					14	3	foo2.adb
FOO2		2			3	3	foo2.adb
OVFL.1		1			7	17	conversion from -2**63..2**63-1 to -2*
FOOBAR		2		4	232	3	validation 1.adb
SUMMIT		2		1	237	6	validation 1.adb
U-OVFL.1		1			240	23	conversion from 1..3 to VALID.FOOBA
INCR		1		2	176	3	validation 1.adb
U-OVFL.2				1	179	16	conversion from -2**63..2**63-1 to -2*
MAIN	1	8	2	38	370	3	validation 1.adb
U-OVFL.1				1	382	13	conversion from -2**31..2**31-1 to 1..
U-OVFL.5				1	383	15	conversion from -2**63..2**63-1 to -2*
U-OVFL.6				1	390	14	conversion from -2**63..2**63-1 to -2*
U-OVFL.8				1	390	16	conversion from -2**31..2**31-1 to 1..

PolySpace Viewer

- *Exploitation des résultats : code source*