Présentation de l'équipe/projet « ABSTRACTION » de l'INRIA Paris–Rocquencourt commune au CNRS et à l'ENS

Patrick Cousot

Responsable du projet

Patrick.Cousot@ens.fr www.di.ens.fr/~cousot

Comité de Direction de l'INRIA

Paris, 10 juin 2008

Plan

1. Composition de l'équipe-projet « ABSTRACTION »

Composition de l'équipe-projet « Abstraction »

Permanents

Patrick Cousot (ENS)

Bruno Blanchet (CNRS)

Radhia Cousot (CNRS)

Jérôme Feret (INRIA)

Laurent Mauborgne (ENS)

Antoine Miné (CNRS)

Xavier Rival (INRIA)

Doctorants

Julien Bertrane (ENS)

Guillaume Capron (X)

Pietro Ferrara (X)

Ingénieur

Élodie-Jane Sims (CDD ENS)

Post-doc.

Axel Simon (CDD ENS)

Visiteurs (1 à 12 mois en 2007/08)

Thésards : F. Camporesi (Bologne), L. Chen (Changsha) ; Chercheurs : A. Cortesi (Venise), R. Giacobazzi (Vérone), B. Goldberg (New York), N. Jones (Copenhague), F. Ranzato (Padoue), R. Wilhelm (Sarrebruck).

2. Une introduction très informelle à l'interprétation abstraite

Exemples classiques de bugs du calcul en entiers

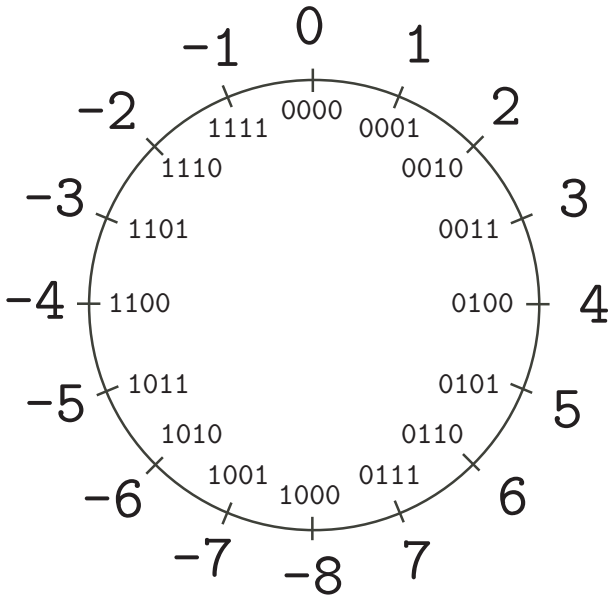
Le programme factorielle (fact.c)

```c
#include <stdio.h>
int fact (int n ) {
  int r, i;
  r = 1;
  for (i=2; i<=n; i++) {
    r = r*i;
  }
  return r;
}
int main() { int n;
  scanf("%d",&n);
  printf("%d!=%d\n",n,fact(n));
}
```

```
% gcc fact.c -o fact.exec
% ./fact.exec
3
3!  = 6
% ./fact.exec
4
4!  = 24
% ./fact.exec
100
100!  = 0
% ./fact.exec
20
20!  = -2102132736
%
```

À la chasse au bug

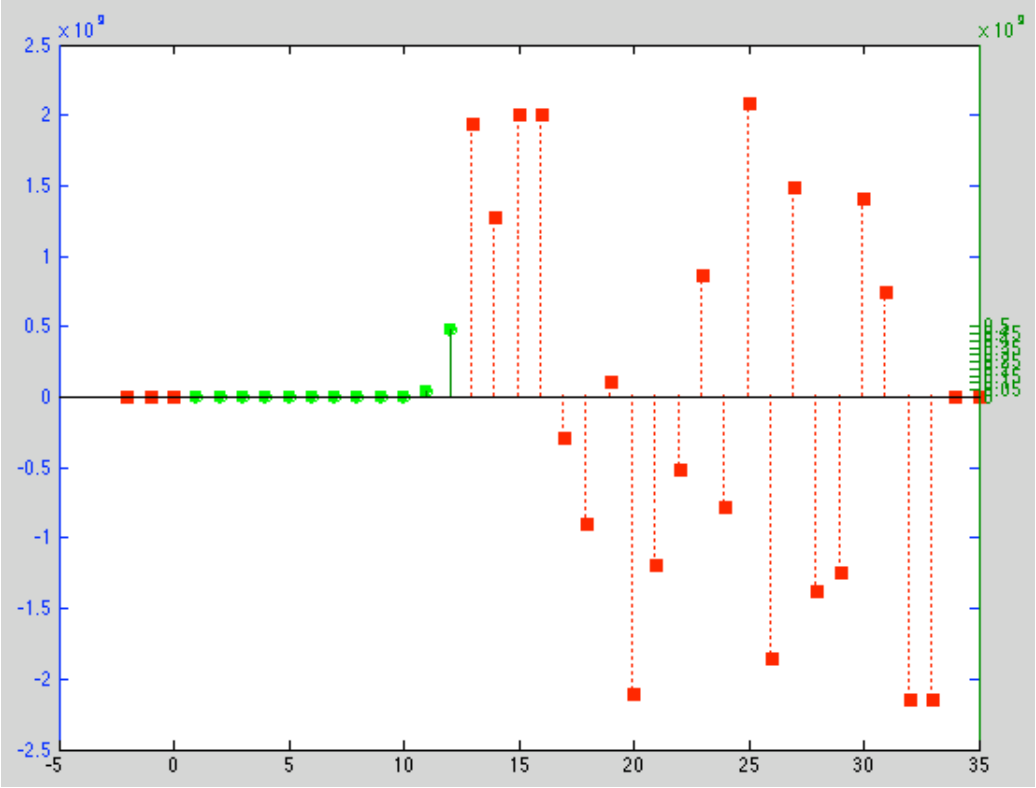
- Les ordinateurs utilisent une arithmétique entière modulaire sur n bits (où $n = 16, 32, 64$, etc)
- Exemple d'une représentation des entiers sur 4 bits (en *complément à deux*) :



- Seuls les entiers entre -8 et 7 sont représentés sur 4 bits
- On obtient $7 + 2 = -7$
  $7 + 9 = 0$

Le bug est une défaillance du programmeur

En machine, la fonction `fact(n)` ne coincide avec $n! = 2 \times 3 \times \ldots \times n$ sur les entiers que pour $1 \leqslant n \leqslant 12$:



Exemples classiques de bugs du calcul en flottants

Les modèles et leur réalisation sur machine

- Les modélisations mathématiques des systèmes physiques utilisent les nombres réels

- Les langages informatiques de modélisation (comme SCADE) utilisent les nombres réels

- Les nombres réels sont difficilement représentables en machine (π a un nombre infini de décimales)

- Les langages informatiques de programmation (comme C ou OCAML) utilisent les nombres flottants

Les flottants

- Les *nombres flottants* sont un sous-ensemble des *rationnels*
- Par exemple on peut représenter 32 flottants sur 6 bits, les 16 flottants positifs étant inégalement répartis comme suit :

- Quand les calculs réels ne tombent pas juste, il faut *arrondir* vers un flottant proche

Exemple d'erreur d'arrondi (1)

$$(x + a) - (x - a) \neq 2a$$

```c
#include <stdio.h>
int main() {
  double x, a; float y, z;
  x = 1125899973951488.0;
  a = 1.0;
  y = (x+a);
  z = (x-a);
  printf("%f\n", y-z);
}
```

```
% gcc arrondi1.c -o arrondi1.exec
% ./arrondi1.exec
134217728.000000
%
```

Exemple d'erreur d'arrondi (2)

$$(x + a) - (x - a) \neq 2a$$

```
#include <stdio.h>
int main() {
  double x, a; float y, z;
  x = 1125899973951487.0;
  a = 1.0;
  y = (x+a);
  z = (x-a);
  printf("%f\n", y-z);
}
```

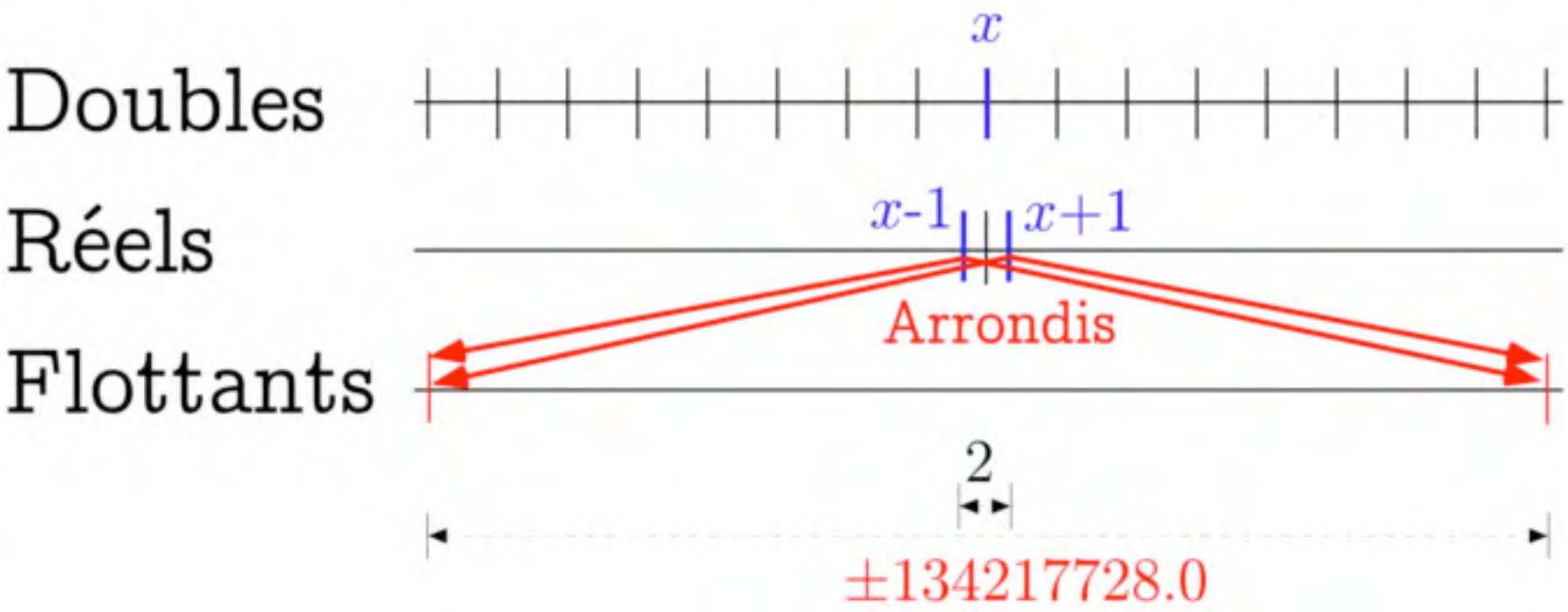
```
% gcc arrondi2.c -o arrondi2.exec
% ./arrondi2.exec
0.000000
%
```

Preuve d'absence d'erreurs à l'exécution par analyse statique

```
% cat -n arrondi3.c
     1 int main() {
     2     double x; float y, z, r;;
     3     x = 1125899973951488.0;
     4     y = x + 1;
     5     z = x - 1;
     6     r = y - z;
     7     __ASTREE_log_vars((r));
     8 }
% astree -exec-fn main -print-float-digits 10 arrondi3.c \
  |& grep "r in "
direct = <float-interval:  r in [-134217728, 134217728] >(1)
```

(1) ASTRÉE considers the worst rounding case (towards $+\infty$, $-\infty$, 0 or to the nearest) whence the possibility to obtain -134217728.

Vérification faite dans le pire des cas



Exemples de bugs dus à des erreurs d'arrondi

- Le bug du missile patriote ratant les Scuds en 1991 à cause une horloge incrémentée par $\frac{1}{10}^{\text{ème}}$ de seconde ($(0,1)_{10} = (0,0001100110011$ en binaire)
- Le bug d'Exel 2007 : 77,1 $\times$ 850 qui donne 65.535 mais s'affiche en 100.000! [2]

2	65535-2^(-37)	100000		65536-2^(-37)	100001
3	65535-2^(-36)	100000		65536-2^(-36)	100001
4	65535-2^(-35)	100000		65536-2^(-35)	100001
5	65535-2^(-34)	65535		65536-2^(-34)	65536
6	65535-2^(-36)-2^(-37)	100000		65536-2^(-36)-2^(-37)	100001
7	65535-2^(-35)-2^(-37)	100000		65536-2^(-35)-2^(-37)	100001
8	65535-2^(-35)-2^(-36)	100000		65536-2^(-35)-2^(-36)	100001
9	65535-2^(-35)-2^(-36)-2^(-37)	65535		65536-2^(-35)-2^(-36)-2^(-37)	65536

(2) Erreur d'arrondi incorrect lors de la traduction de flottants IEEE 754 sur 64 bits en chaîne de caractères Unicode qui conduit à un mauvais alignement dans une table de conversion. Le bug apparaît exactement pour six nombres entre 65534.99999999995 et 65535 et six entre 65535.99999999995 et 65536.

Les bugs dans le monde numérisé quotidien

Les bugs sont fréquents dans la vie quotidienne

– Les bugs se trouvent dans les banques, les voitures, les téléphones, les machines à laver, ...

– Exemple (bug dans un distributeur de monnaie au 19 Boulevard Sébastopol à Paris, le 21 novembre 2006 à $8^{h}30$):

La vérification des programmes

Principe de la vérification des programmes

- Définir une sémantique du langage (c'est-à-dire l'effet de l'exécution des programmes du langage)

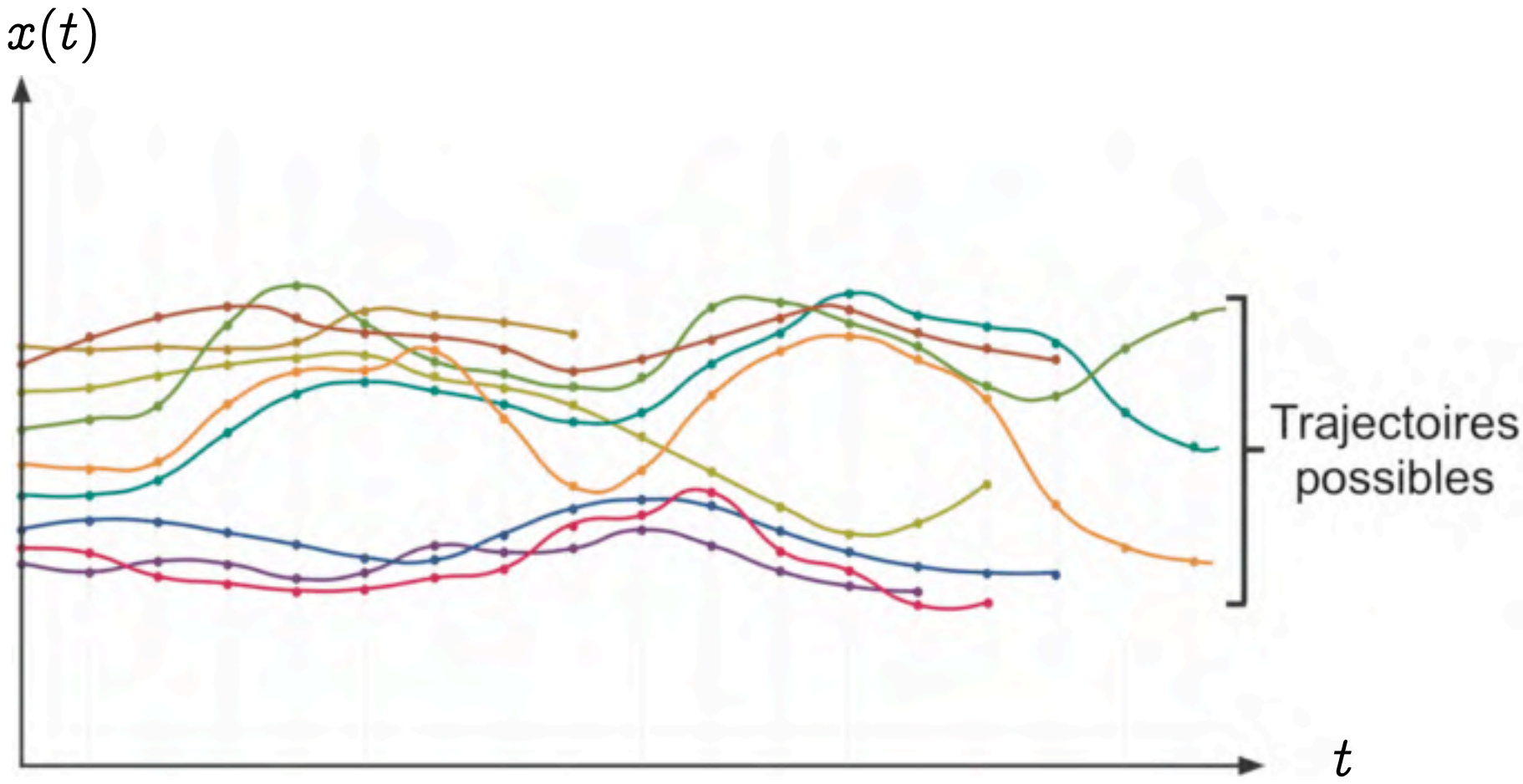
- Définir une spécification (exemple : pas d'erreur à l'exécution comme une division par zéro, un débordement arithmétique, etc)

- Faire une preuve formelle que la sémantique satisfait la spécification

- Utiliser l'ordinateur pour automatiser la preuve

Sémantique des programmes

Sémantique opérationnelle du programme P



Sémantique$[\![P]\!]$

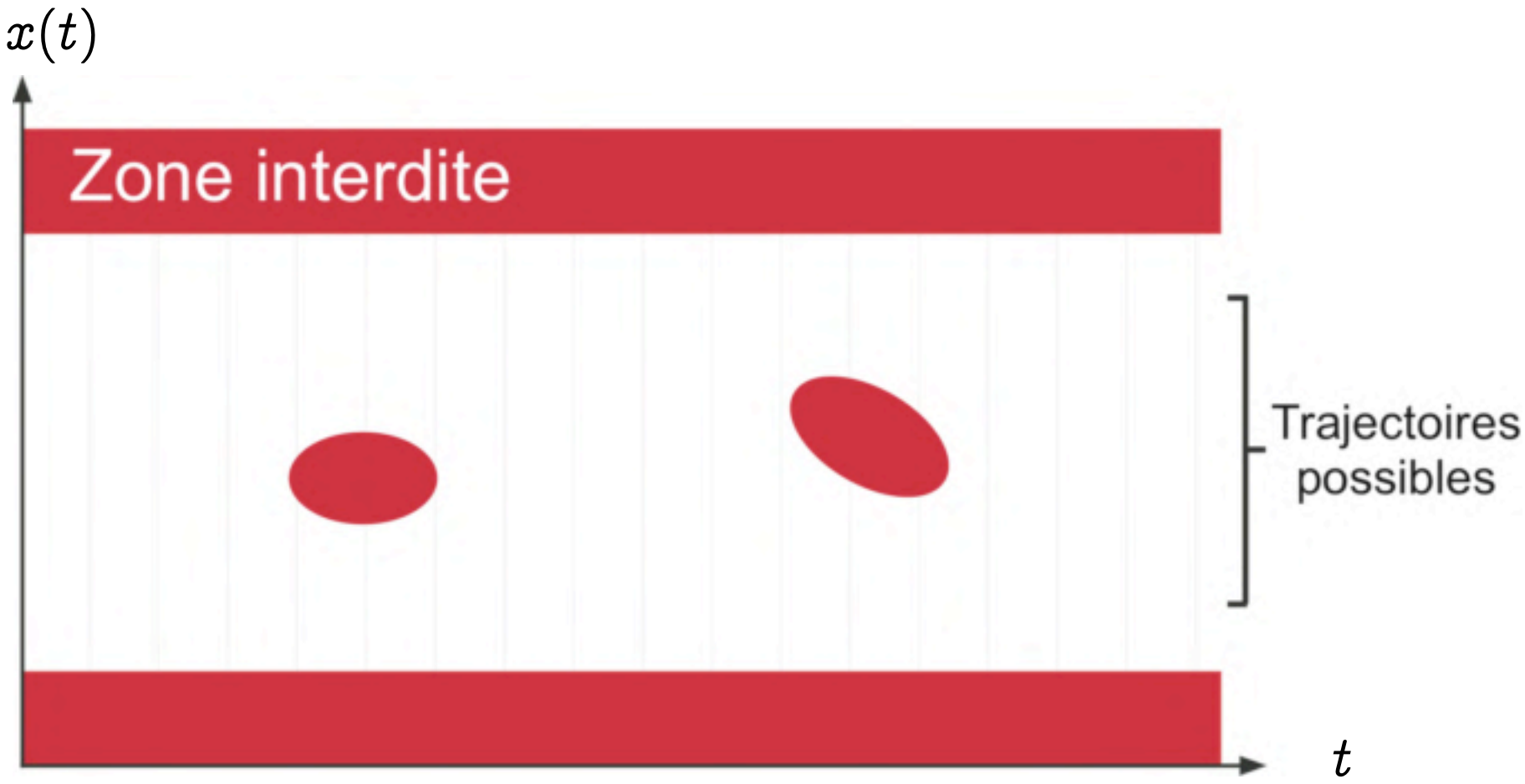
Exemple : trace d'exécution de `fact(4)`

```
int fact (int n ) {
  int r = 1, i;
  for (i=2; i<=n; i++) {
    r = r*i;
  }
  return r;
}
```

- $n \leftarrow 4;\ r \leftarrow 1;$
- $i \leftarrow 2;\ r \leftarrow 1 \times 2 = 1;$
- $i \leftarrow 3;\ r \leftarrow 2 \times 3 = 6;$
- $i \leftarrow 4;\ r \leftarrow 6 \times 4 = 24;$
- $i \leftarrow 5;$
- return 24;

Spécification des programmes

Spécification du programme P



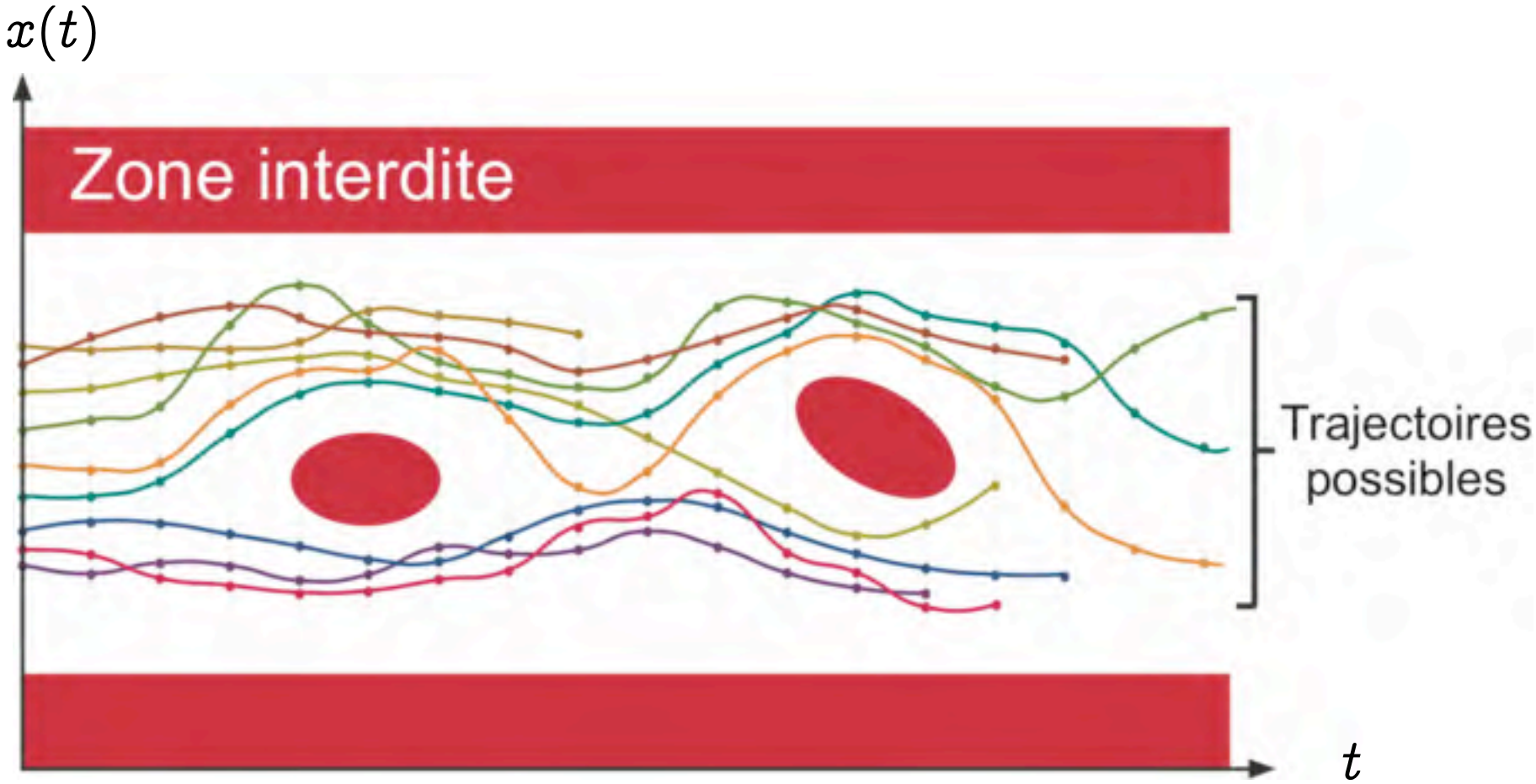
Spécification$[\![P]\!]$

Exemple de spécification

```
int fact (int n ) {
  int r, i;
  r = 1;
  for (i=2; i<=n; i++) {     ← pas de débordement de i++
    r = r*i;                 ← pas de débordement de r*i
  }
  return r;
}
```

Preuve formelle

Preuve formelle du programme P



$$\text{Sémantique}[\![P]\!] \subseteq \text{Spécification}[\![P]\!]$$

Indécidabilité et complexité

- Le problème de la preuve mathématique formelle est indécidable [3]

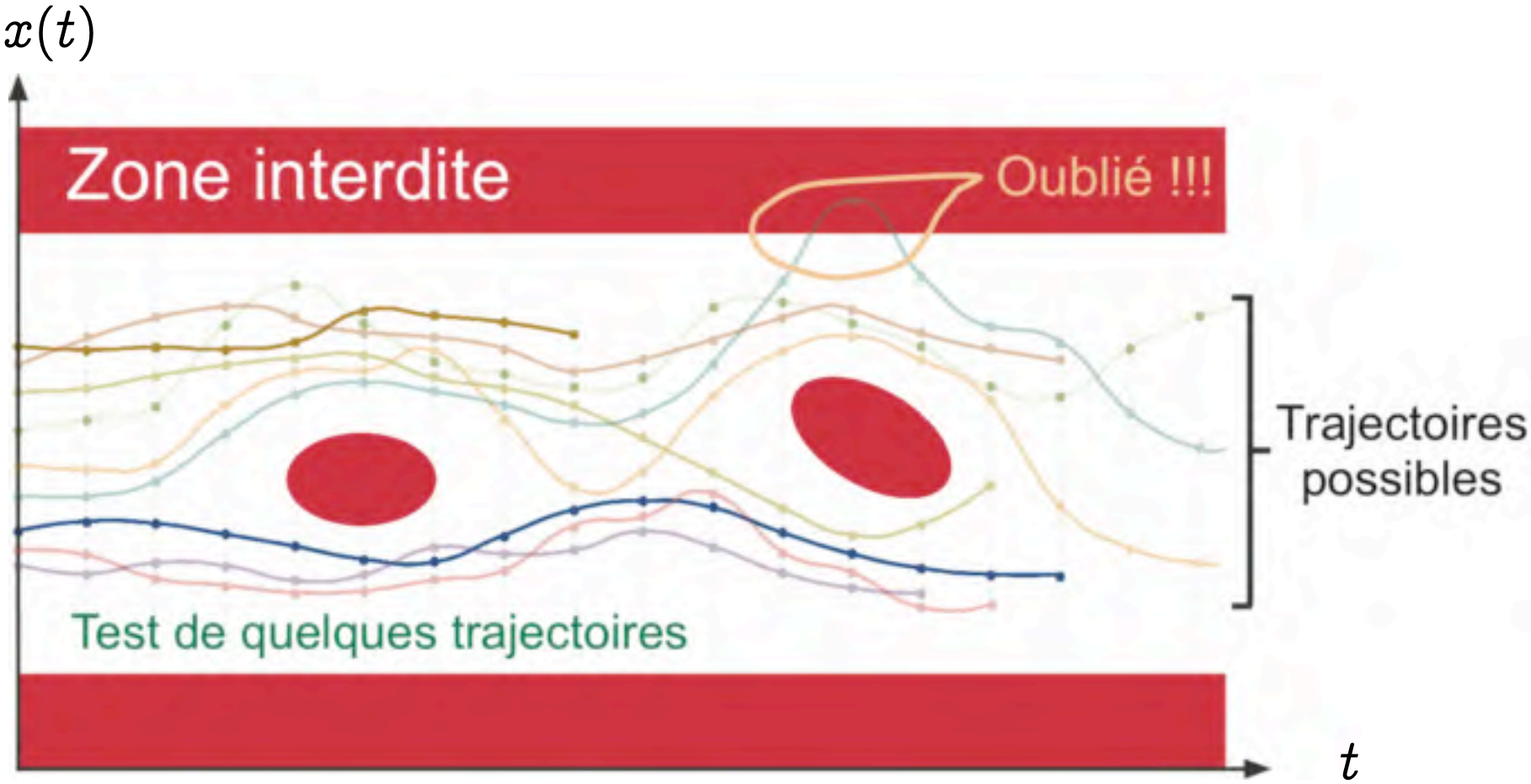
- Même en supposant tout fini, la complexité est beaucoup trop élevée

- Exemple: 1.000.000 lignes $\times$ 50.000 variables $\times$ 64 bits $\simeq 10^{27}$ états

- À raison de l'examen de 10^{15} états par seconde, il faudrait 10^{12} s $>$ 300 siècles (et beaucoup de mémoire) !

[3] un ordinateur ne peut pas toujours le résoudre en un temps fini.

Le test est incomplet

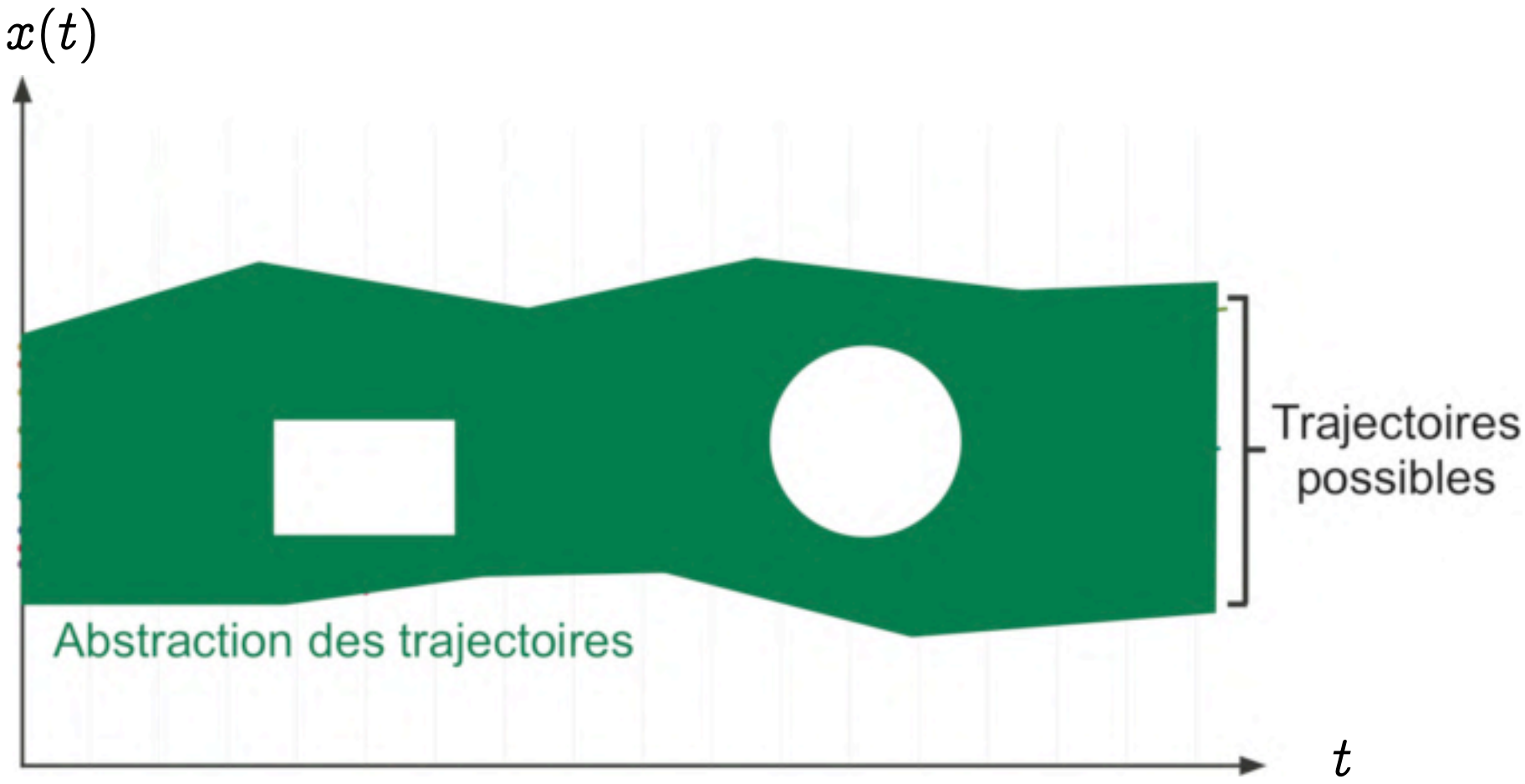


Interprétation abstraite [1]

Référence

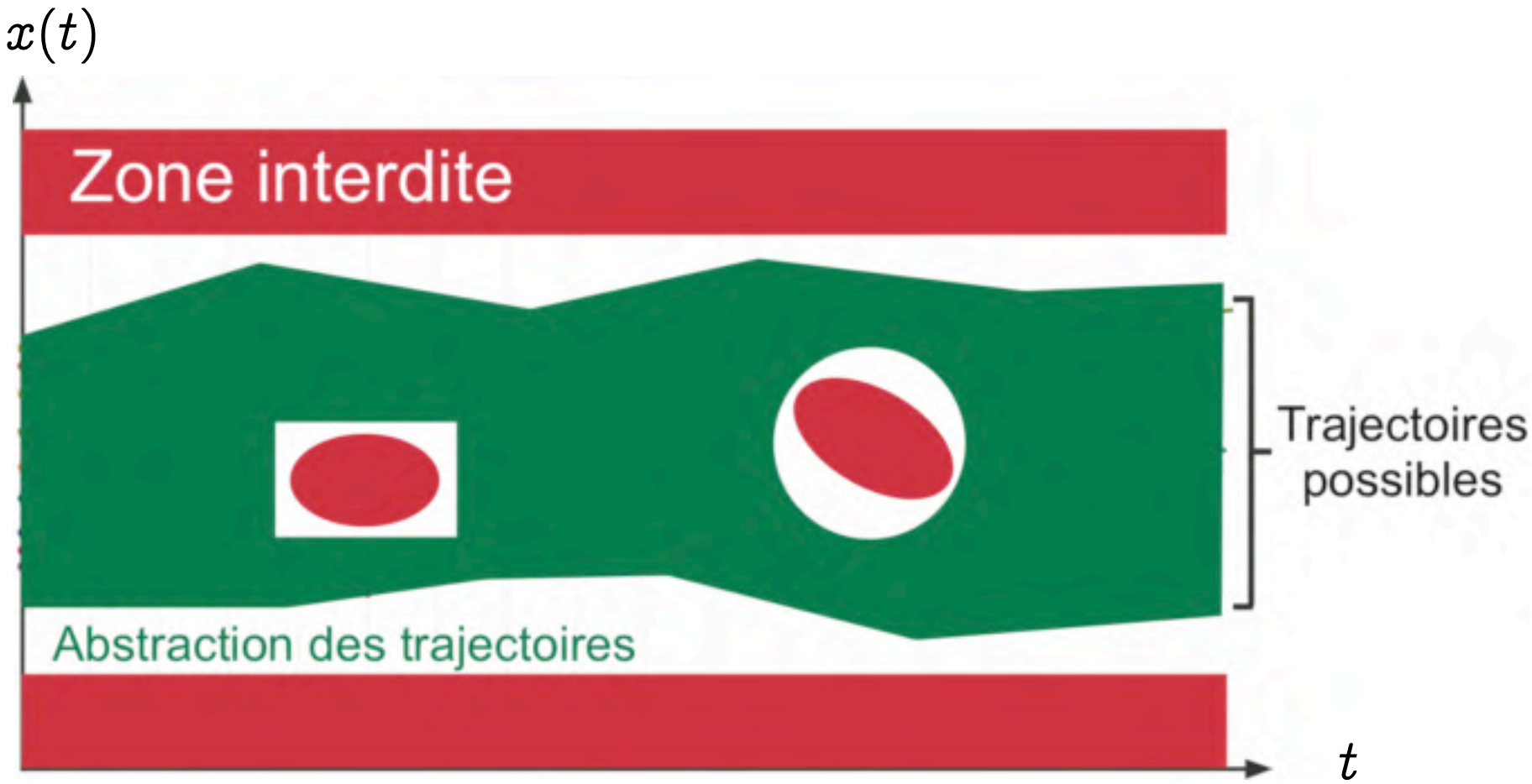
[1] P. Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes. Thèse d'État ès sciences mathématiques. Université scientifique et médicale de Grenoble. 1978.

Abstraction du programme P



Abstraction(Sémantique$[\![P]\!]$)

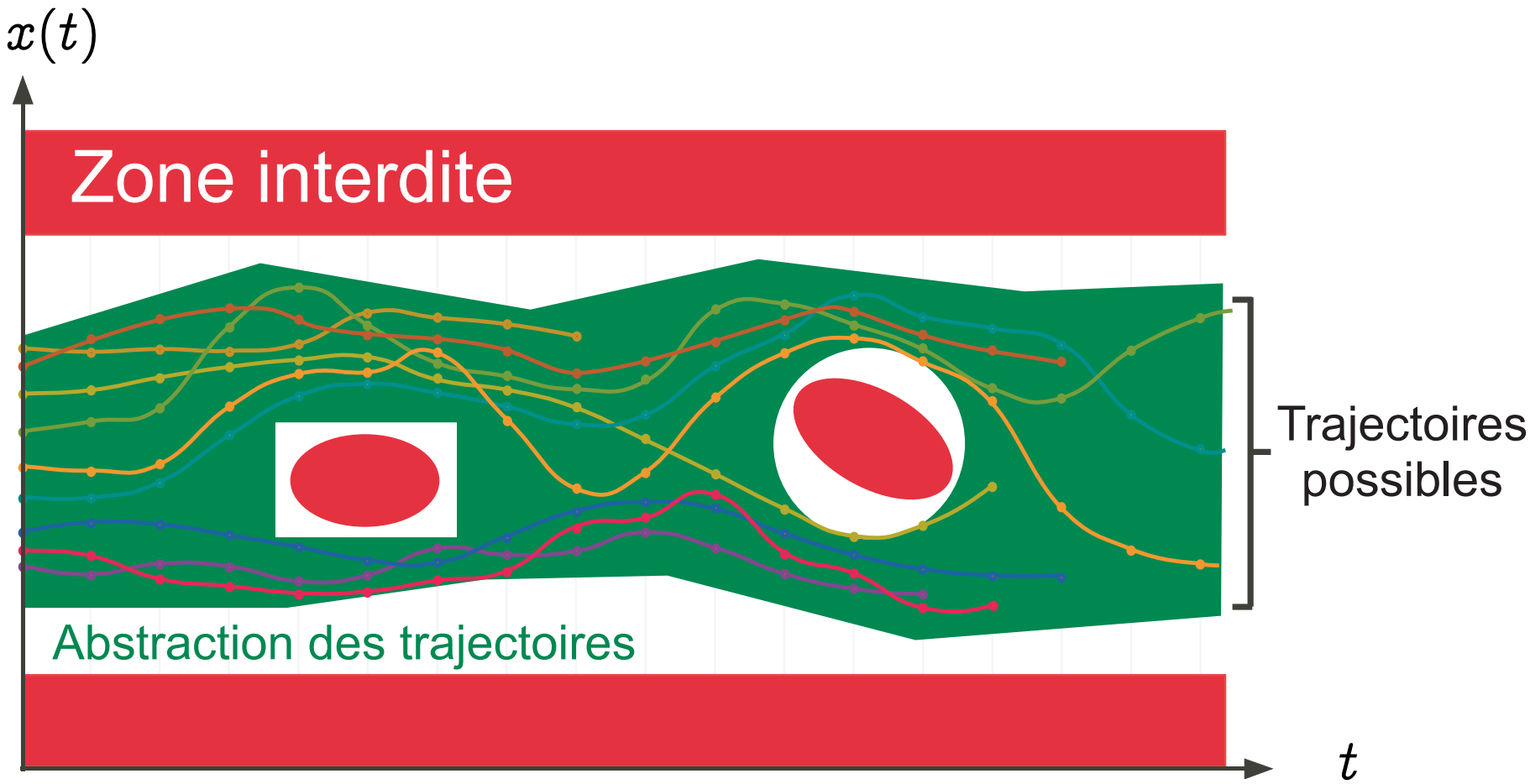
Preuve par abstraction



$$\text{Abstraction}(\text{Sémantique}[\![P]\!]) \subseteq \text{Spécification}[\![P]\!]$$

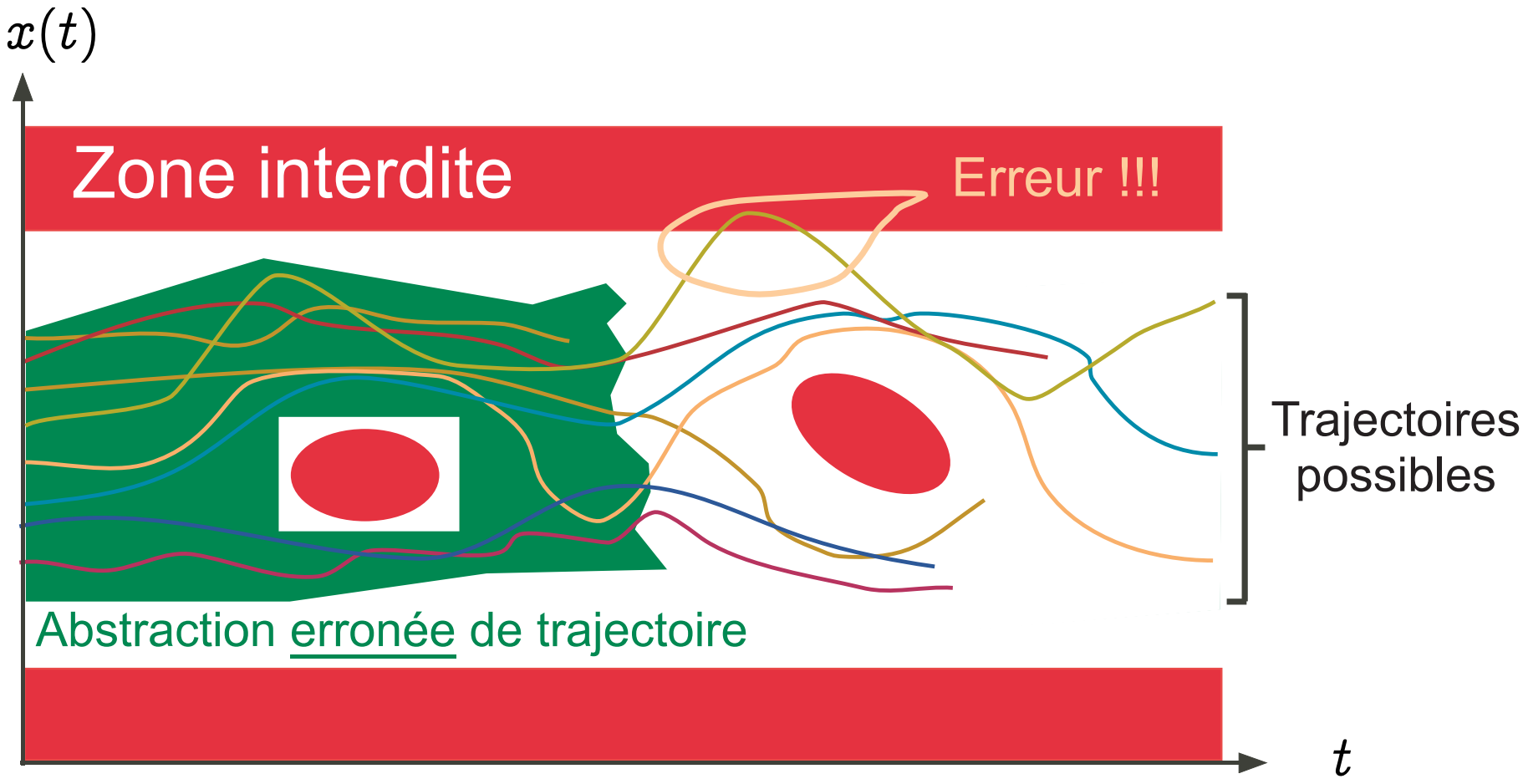
Correction de l'interprétation abstraite

L'interprétation abstraite est correcte



$$\text{Sémantique}[\![P]\!] \subseteq \text{Abstraction}(\text{Sémantique}[\![P]\!])$$

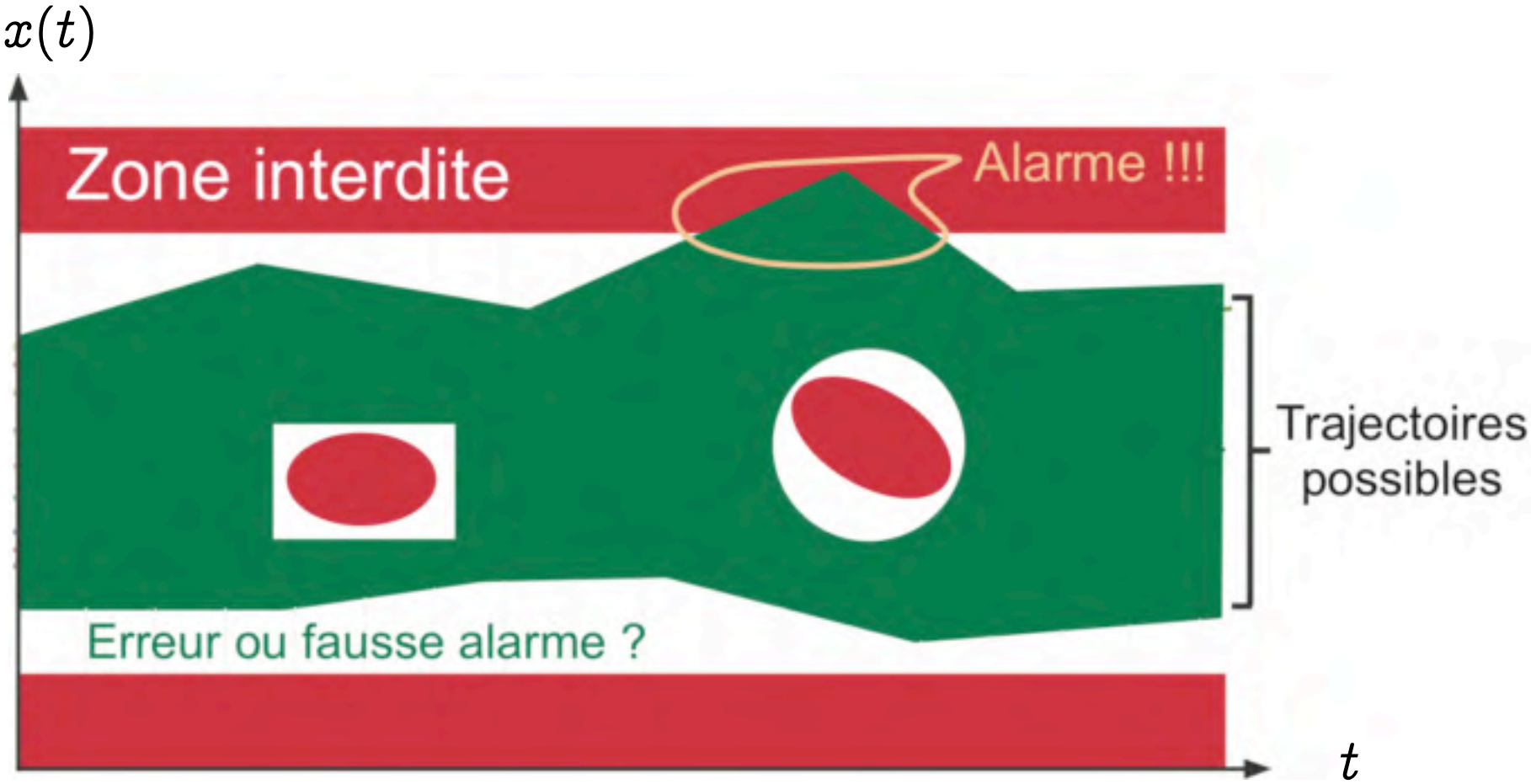
Les abstractions erronées ne permettent pas de conclure valablement (faux négatifs) [4]



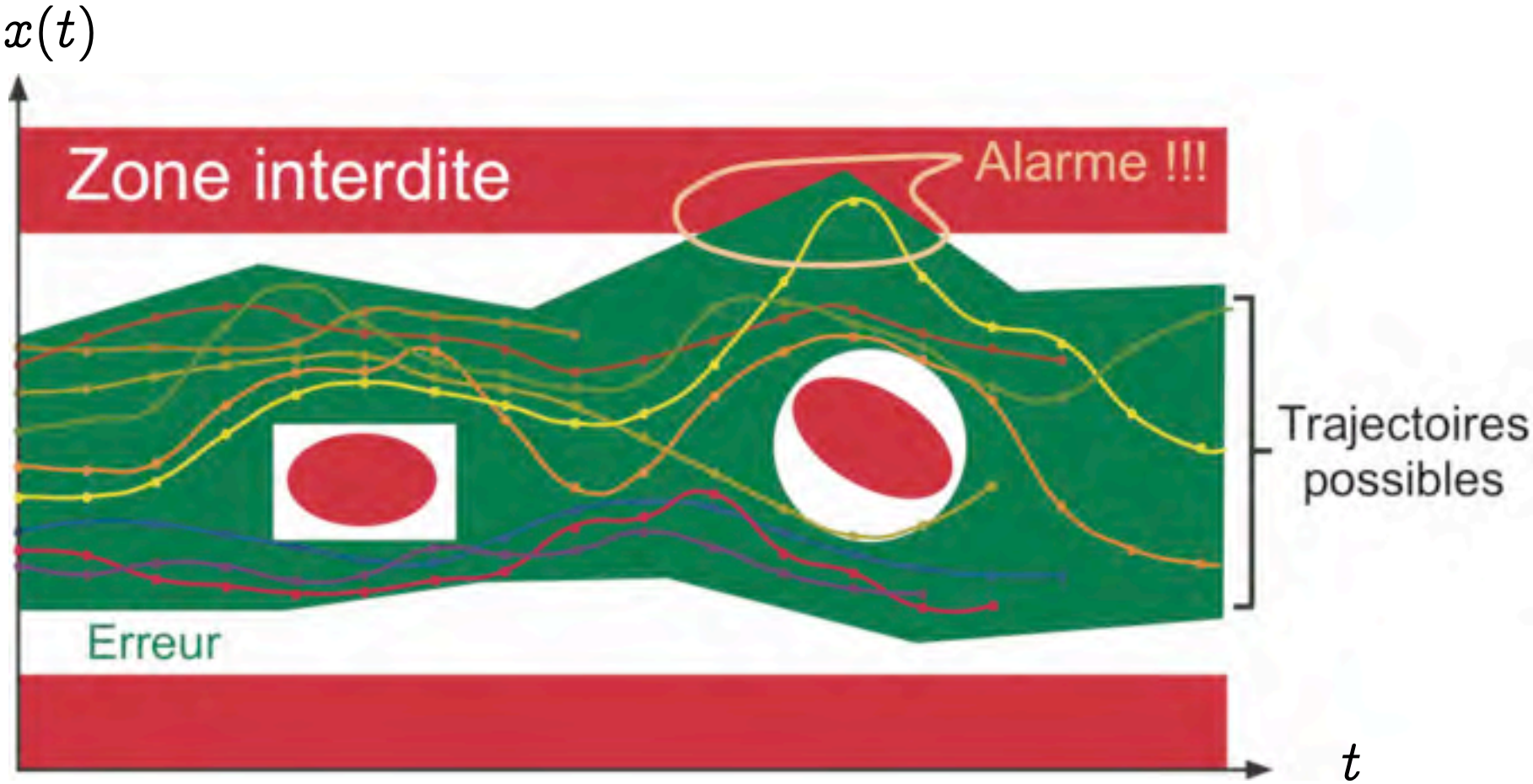
[4] Cette situation est toujours exclue par la théorie de l'interprétation abstraite.

Incomplétude de l'interprétation abstraite

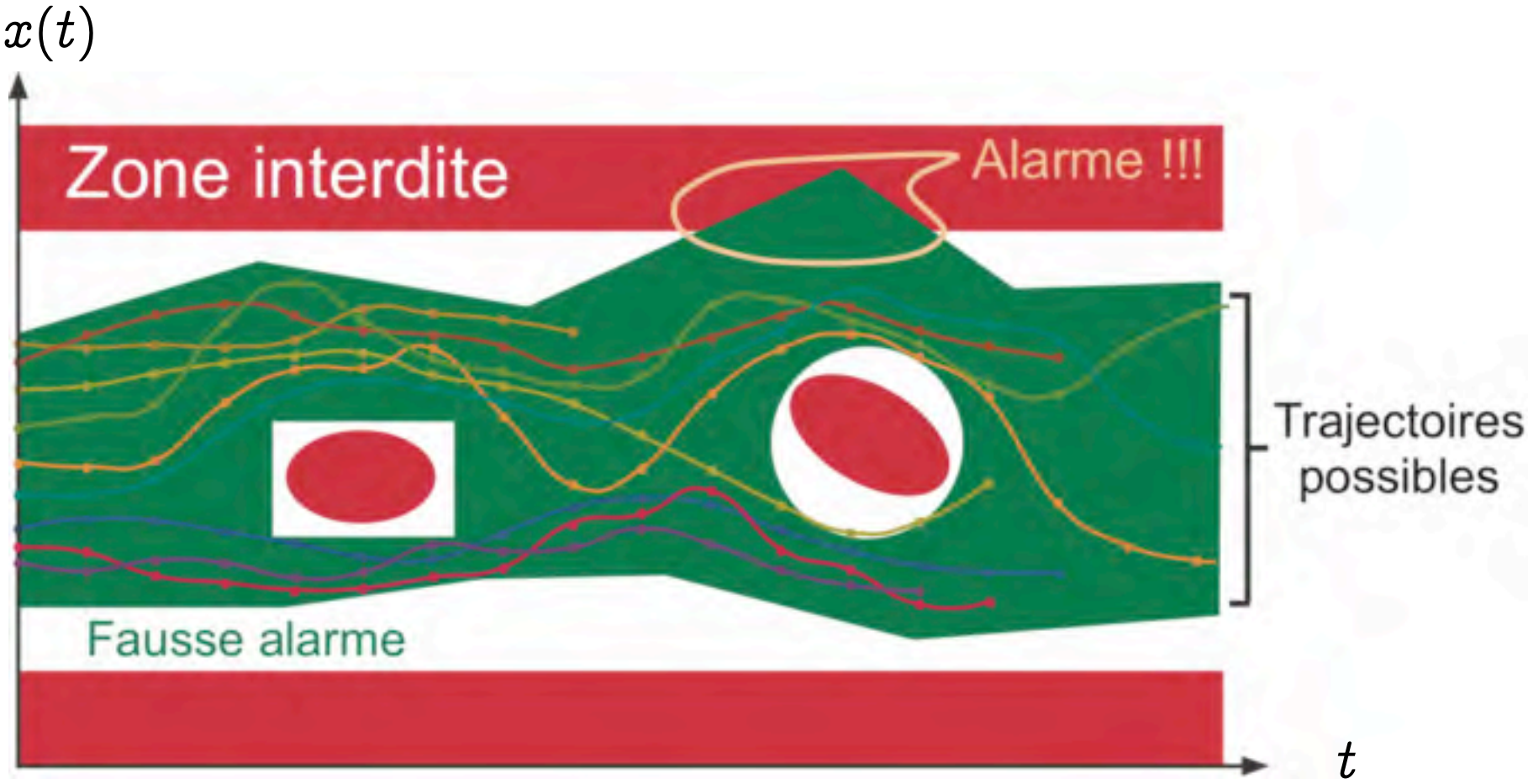
Alarme


$x(t)$
Zone interdite
Alarme !!!
Trajectoires possibles
Erreur ou fausse alarme ?
t

Une alarme peut correspondre à une erreur



Une alarme peut correspondre à une approximation (faux positif)



Applications théoriques de l'interprétation abstraite

Applications de l'interprétation abstraite

- L'analyse statique [CC77], [CH78], [CC79] y compris « dataflow analysis » [CC79], [CC00], « set-based analysis » [CC95], « predicate abstraction » [Cou03], ...
- L'analyse de grammaires et l'analyse syntaxique [CC03];
- Les hiérarchies de sémantiques et de méthodes de preuves [CC92], [Cou02];
- Le typage et l'inférence de types [Cou97];
- Le « model checking » abstrait [CC00];
- Les transformations de programmes (y compris l'optimisation des programmes, l'évaluation partielle, etc) [CC02];
- Le « software watermarking » [CC04];

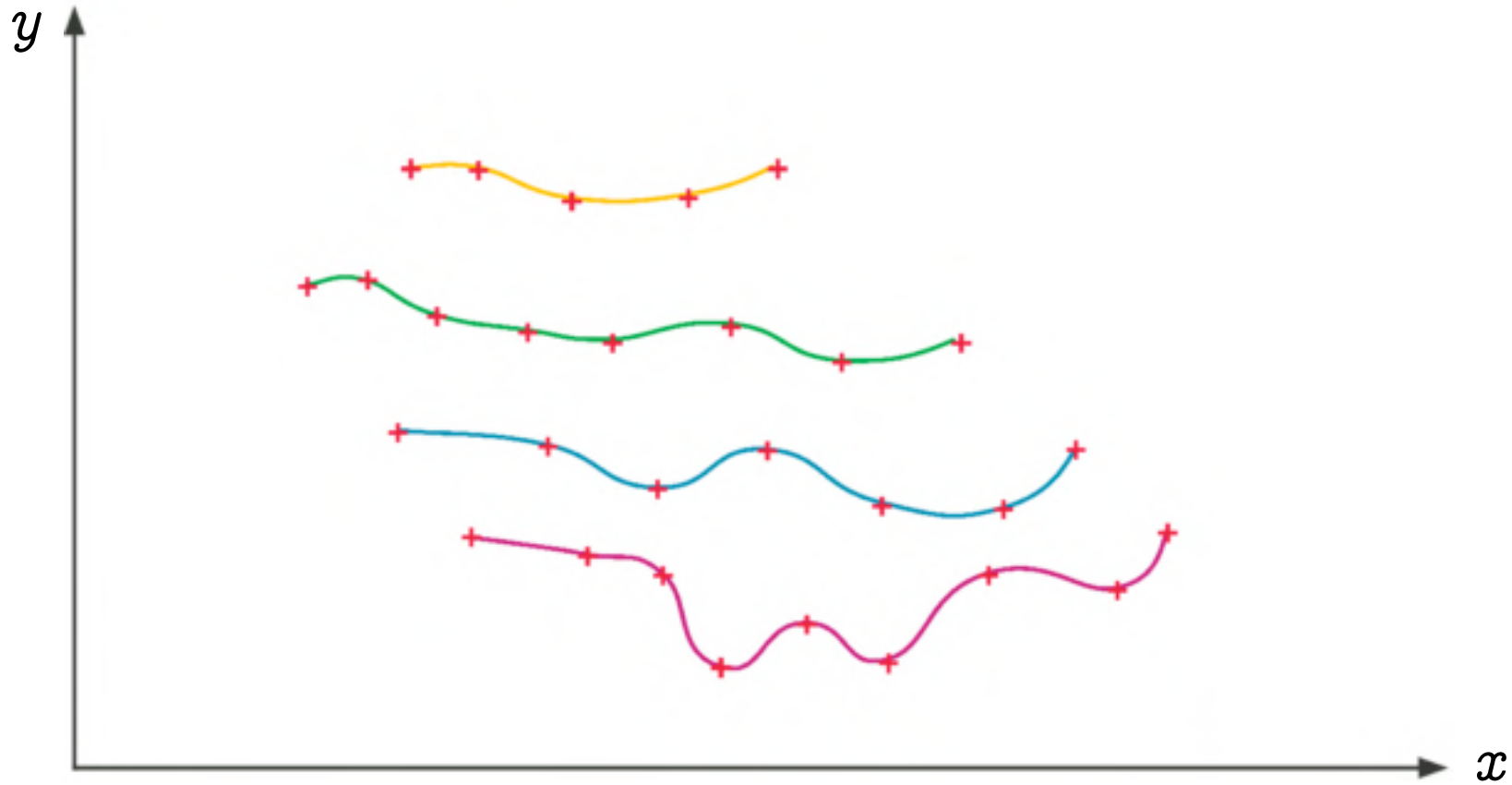
Applications de l'interprétation abstraite (Suite)

- Les bisimulations [RT04, RT06];
- La sécurité basée sur les languages [GM04];
- La vérification de protocoles cryptographiques dans le modèle formel [Bla05];
- la détection sémantique de « malware » obscurci [PCJD07].
- Les bases de données [AGM93, BPC01, BS97]
- La « computational biology » [DFFK07, DFF+07, DFFK08, Fer07]
- Le calcul quantique [JP06, Per06]

Toutes ces techniques mettent en œuvre des approximations sûres qui peuvent être formalisées par l'interprétation abstraite.

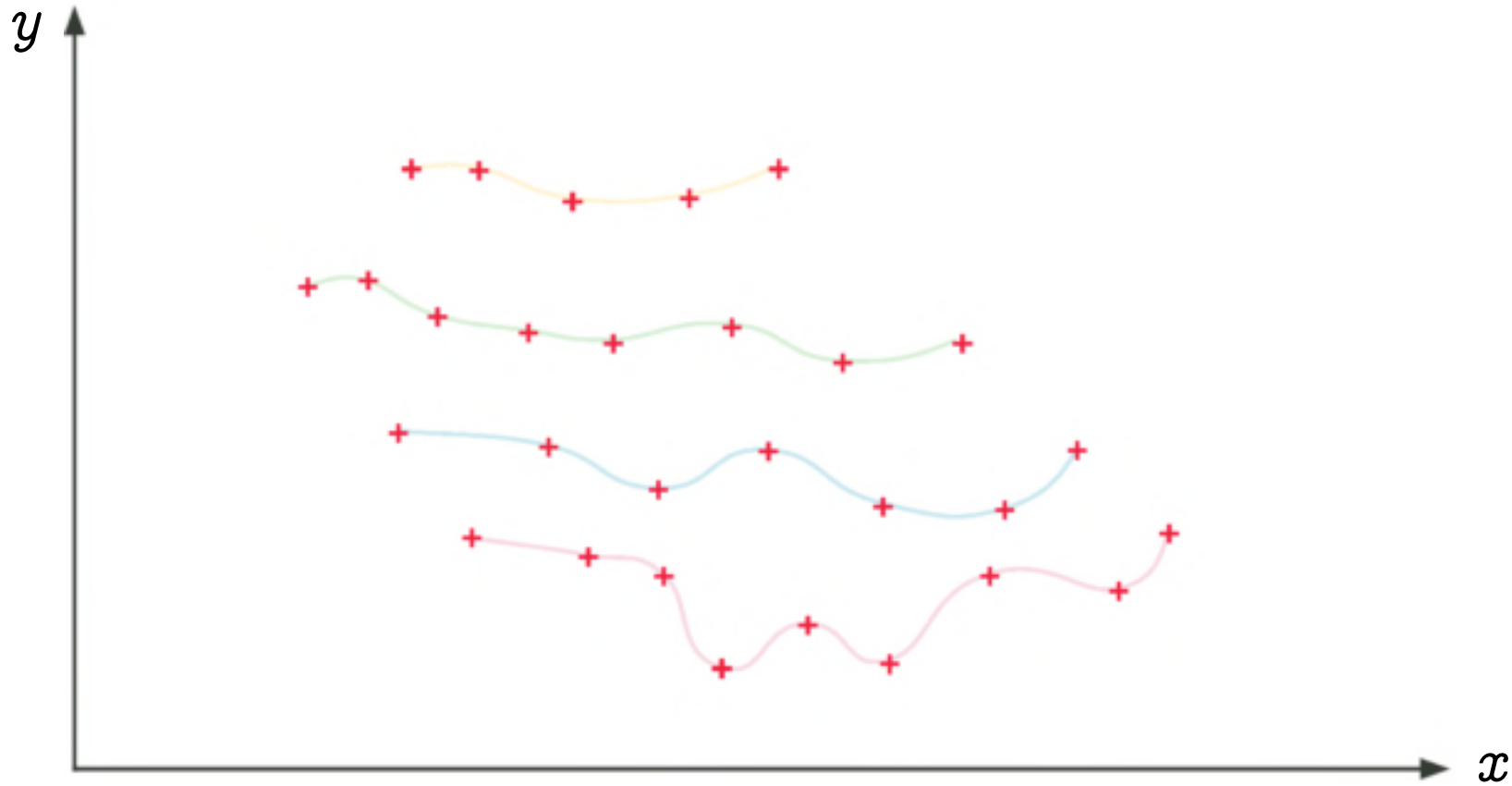
Application de l'interprétation abstraite à l'analyse statique

Sémantique



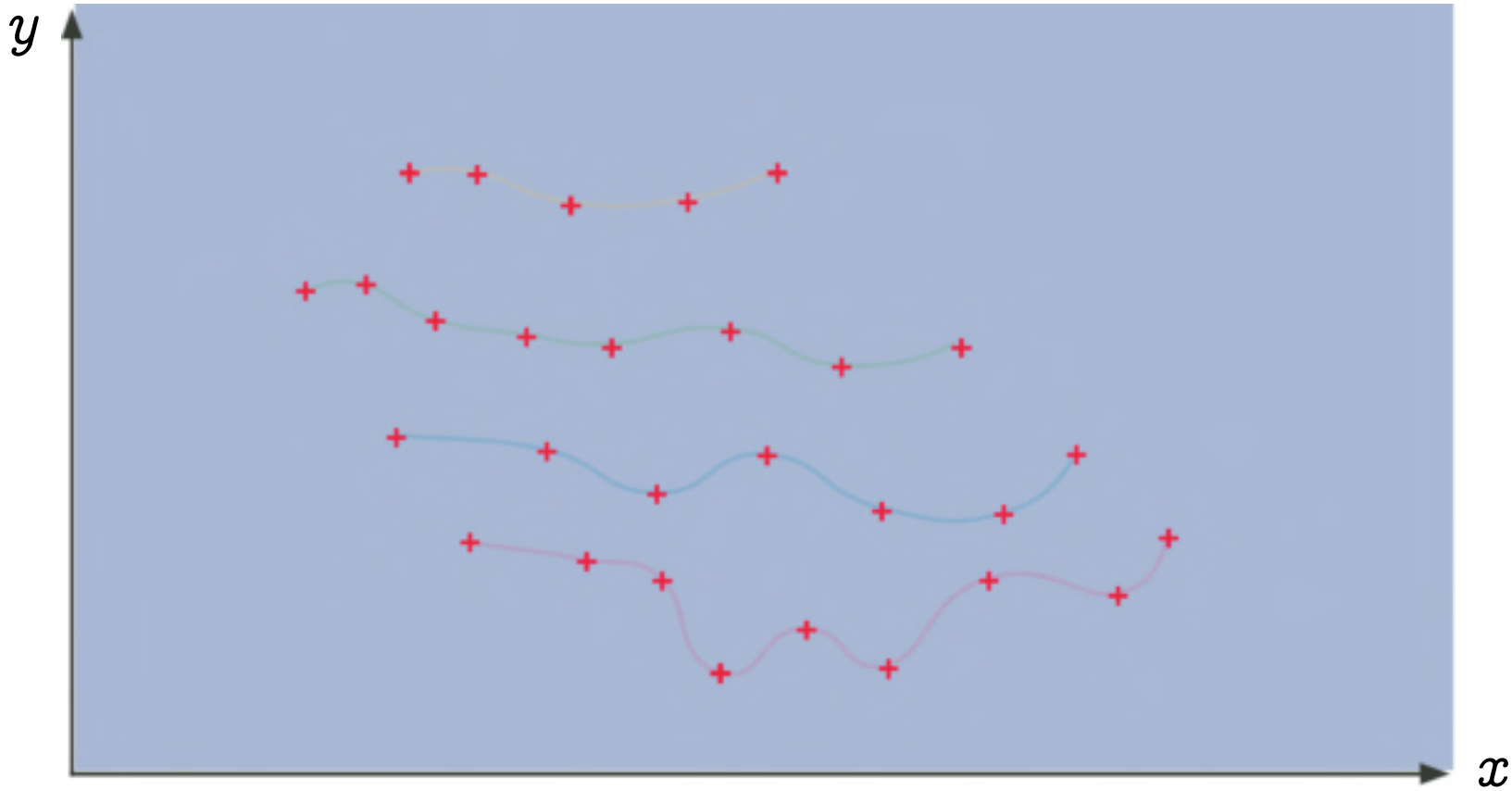
Ensemble (infini) de traces (finies ou infines)

Abstraction en un ensemble d'états (invariant)



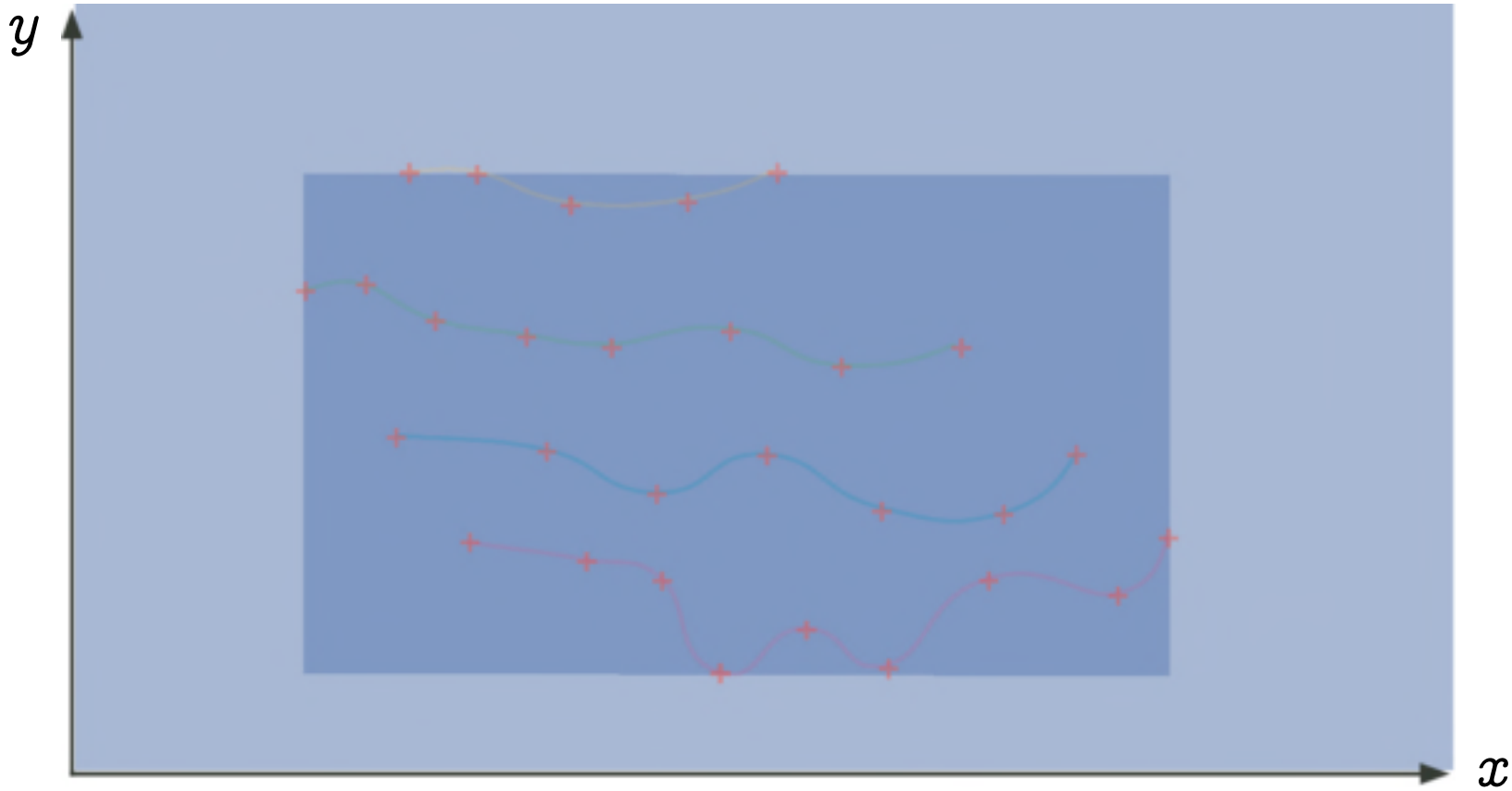
Ensemble de points $\{(x_i, y_i) : i \in \Delta\}$, Hoare logic [Cou02]

Abstraction par des signes



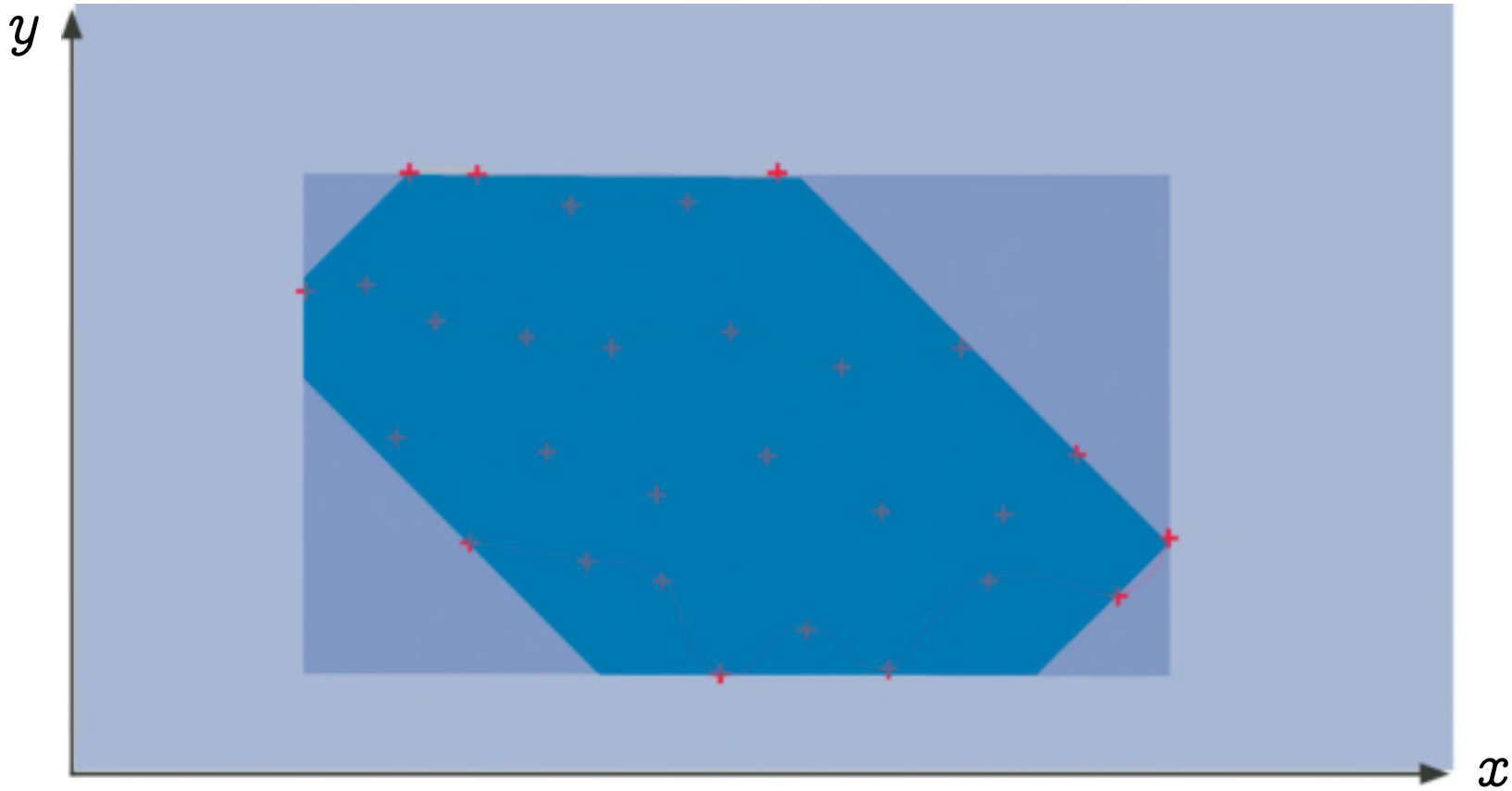
Signes $x \geq 0$, $y \geq 0$ [CC79]

Abstraction par des intervalles



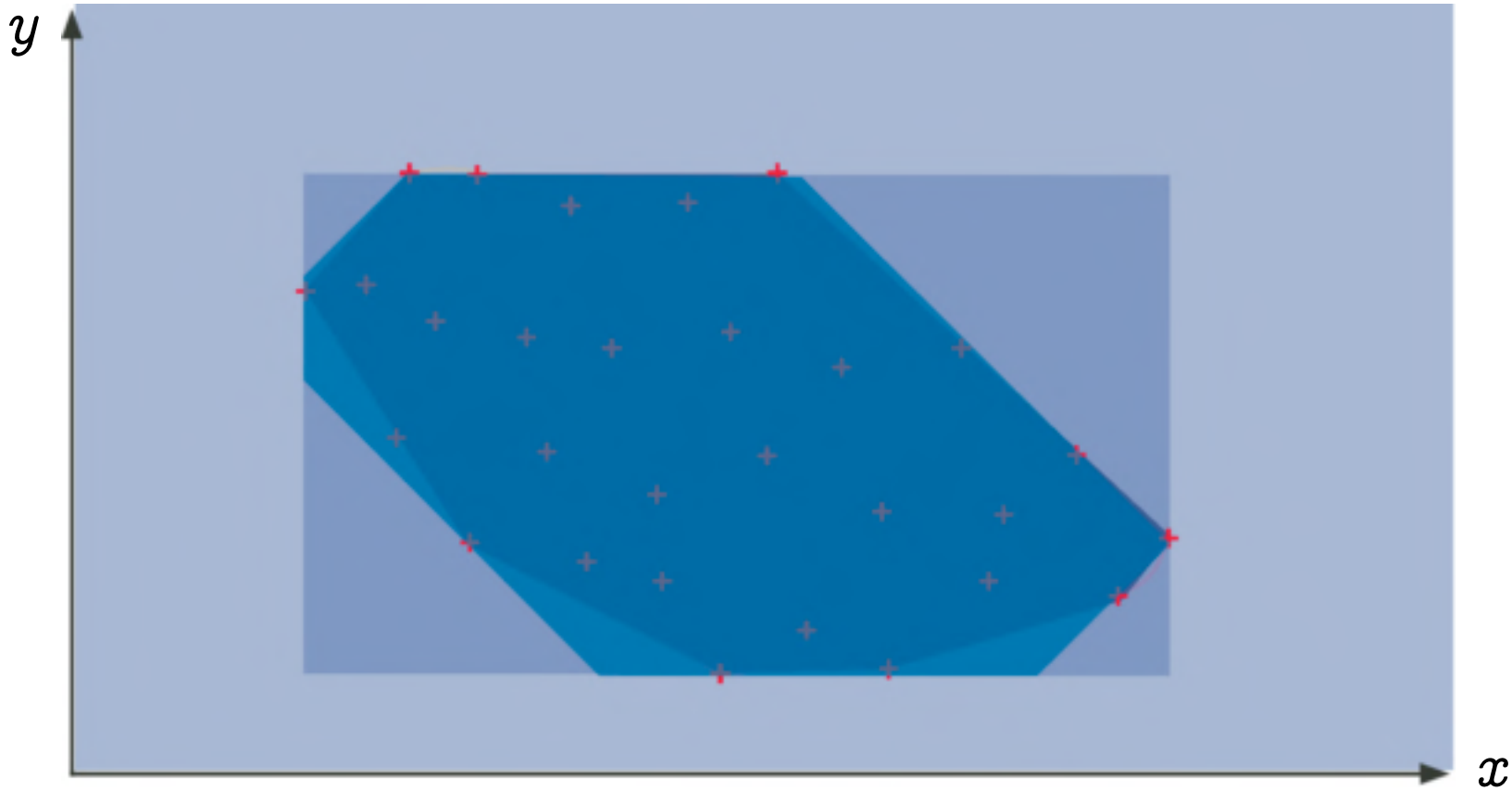
Intervalles $a \leq x \leq b$, $c \leq y \leq d$ [CC77]

Abstraction par des octogones



Octogones $x - y \leq a$, $x + y \leq b$ [Min06]

Abstraction par des polyèdres



Polyèdres $a.x + b.y \leq c$ [CH78]

Calcul d'invariant par approximation de points fixes [CC77]

Équation de point fixe

$\{y \geqslant 0\}$ $\leftarrow$ hypothèse

```
x = y
```

$\{I(x,y)\}$ $\leftarrow$ invariant de boucle

```
while (x > 0) {
  x = x - 1;
}
```

Conditions de vérification de Floyd-Hoare :

$$(y \geqslant 0 \wedge x = y) \Longrightarrow I(x,y) \qquad \textit{initialisation}$$

$$(I(x,y) \wedge x > 0 \wedge x' = x - 1) \Longrightarrow I(x',y) \qquad \textit{iteration}$$

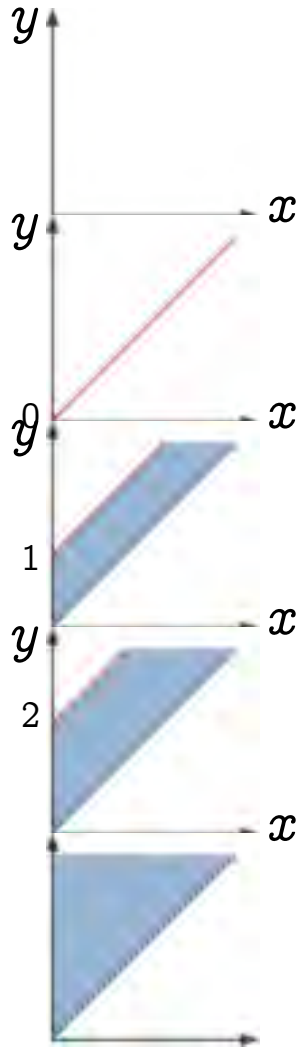
Équation de point fixe :

$$I(x,y) \;=\; x \geqslant 0 \wedge (x = y \vee I(x+1,y)) \qquad (\textit{i.e. } I = F(I)^{(5)})$$

(5) On cherche l'invariant I le plus précis, qui implique tous les autres

Itérés $I = \lim_{n\to\infty} F^n(\text{faux})$ ... accélérés

$$
\begin{aligned}
I^0(x,y) &= \text{faux} \\
I^1(x,y) &= x \geqslant 0 \wedge (x = y \vee I^0(x+1,y)) \\
&= 0 \leqslant x = y \\
I^2(x,y) &= x \geqslant 0 \wedge (x = y \vee I^1(x+1,y)) \\
&= 0 \leqslant x \leqslant y \leqslant x+1 \\
I^3(x,y) &= x \geqslant 0 \wedge (x = y \vee I^2(x+1,y)) \\
&= 0 \leqslant x \leqslant y \leqslant x+2 \\
I^4(x,y) &= I^2(x,y) \nabla I^3(x,y) \leftarrow \text{élargissement} \\
&= 0 \leqslant x \leqslant y \\
I^5(x,y) &= x \geqslant 0 \wedge (x = y \vee I^4(x+1,y)) \\
&= I^4(x,y) \quad \text{point fixe!}
\end{aligned}
$$



Tous ces calculs peuvent être faits avec des octogones!

Passage à l'échelle

Difficulté du passage à l'échelle

- Analyse efficace (avec des ressources de calcul raisonables) $\Rightarrow$ abstraction grossière

- Pas de fausses alarmes $\Rightarrow$ abstraction précise

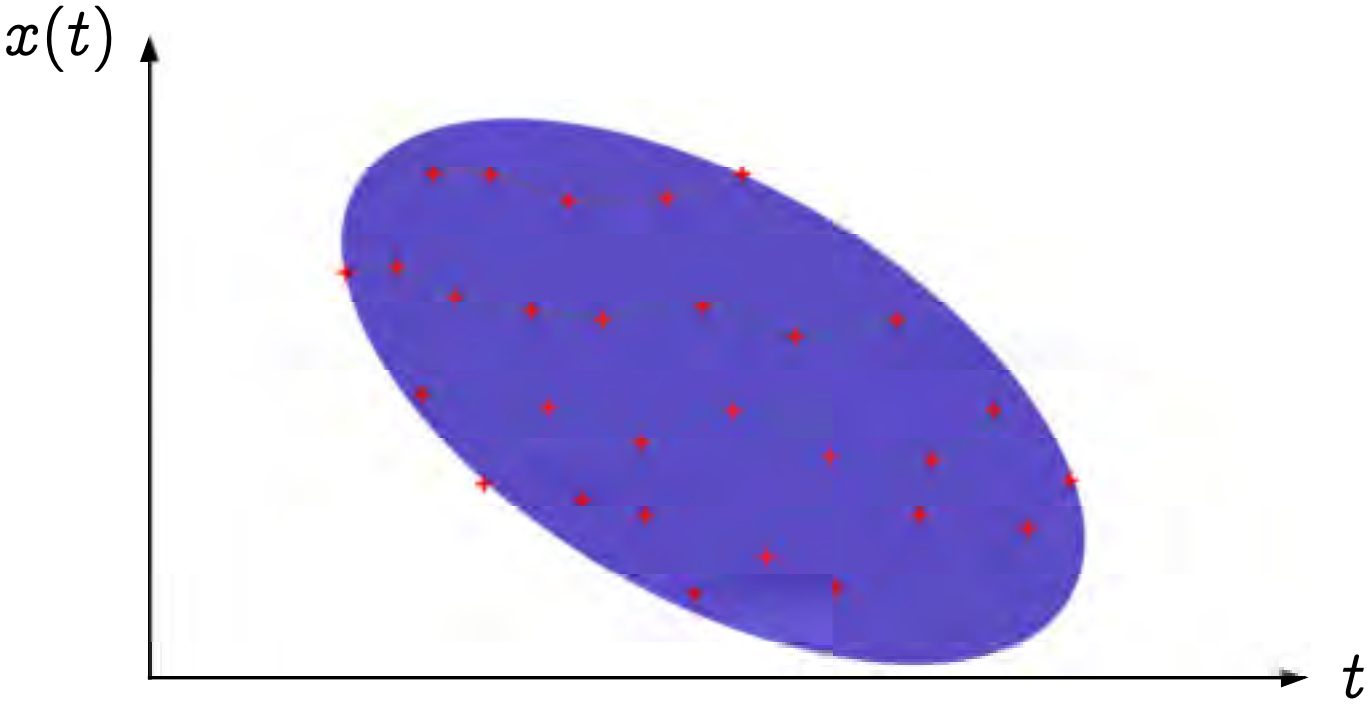
Ajustement du ratio coût/précision dans ASTRÉE

- Combinaison d'abstractions

- Abstractions spécialisées pour le domaine d'application visé

- Ajustement du ratio coût/précision des abstractions :
  - Globalement par paramétrisation
  - Localement par l'ajout (automatique) de directives d'analyse

Abstraction par des ellipsoïdes pour les filtres

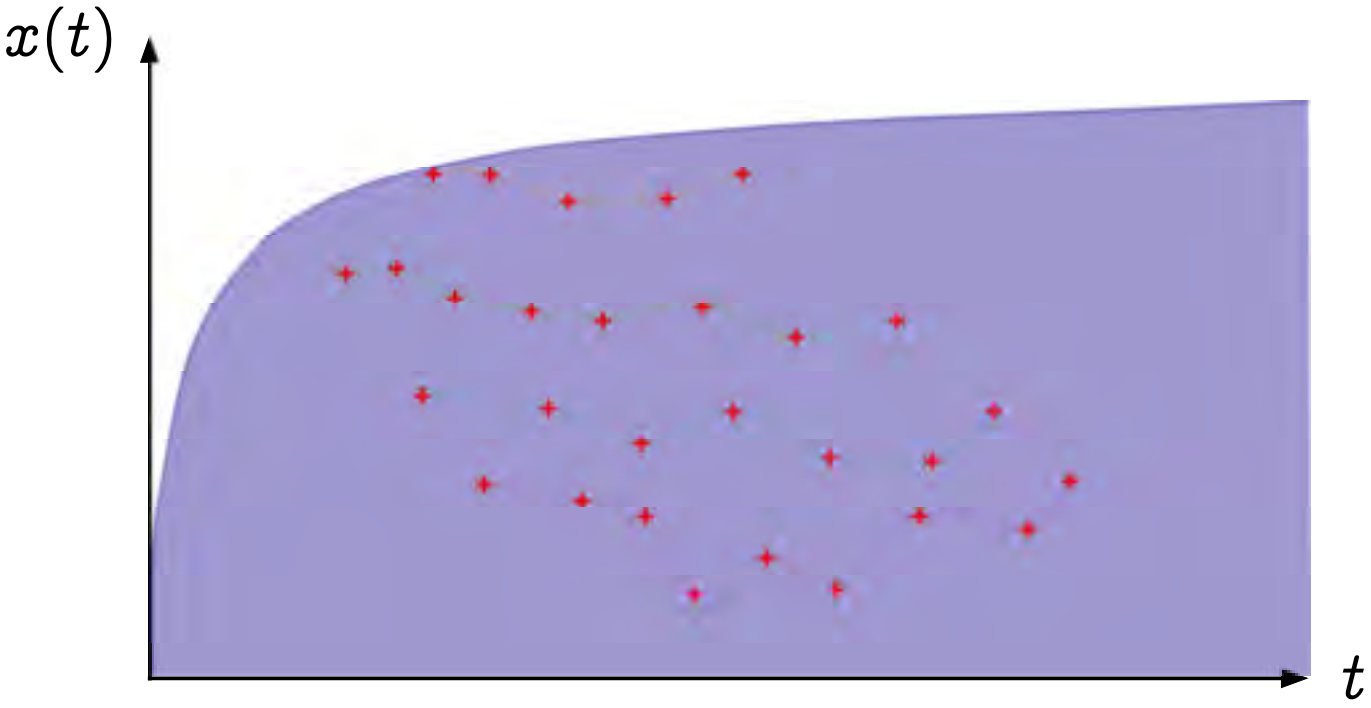


Ellipsoïdes $(x-a)^2 + (y-b)^2 \leq c$ [Fer05b]

Exemple d'analyse de filtre par ASTRÉE

```
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
             + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```

Abstraction par des exponentielles pour l'accumulation de petites erreurs d'arrondi



Exponentielles $a^x \leq y$ [Fer05a]

Exemple d'analyse par ASTRÉE

```
% cat retro.c
typedef enum {FALSE=0, TRUE=1} BOOL;
BOOL FIRST;
volatile BOOL SWITCH;
volatile float E;
float P, X, A, B;

void dev( )
{ X=E;
  if (FIRST) { P = X; }
  else
    { P =  (P - ((((2.0 * P) - A) - B)
            * 4.491048e-03)); };
  B = A;
  if (SWITCH) {A = P;}
  else {A = X;}
}
```

```
void main()
{ FIRST = TRUE;
  while (TRUE) {
    dev( );
    FIRST = FALSE;
    __ASTREE_wait_for_clock(());
  }}
% cat retro.config
__ASTREE_volatile_input((E [-15.0, 15.0]));
__ASTREE_volatile_input((SWITCH [0,1]));
__ASTREE_max_clock((3600000));
```

```
|P| <= (15.  + 5.87747175411e-39
/ 1.19209290217e-07) * (1 +
1.19209290217e-07)^clock - 5.87747175411e-39
/ 1.19209290217e-07 <= 23.0393526881
```

Applications industrielles de l'interprétation abstraite

Exemples d'analyseurs statiques à usage industriel

– Pour les programmes C critiques synchrones embarqués de contrôle/commande (par exemple pour des logiciels de Commande De Vol Électrique)

– aiT [FHL+01] est un analyseur statique qui vérifie les temps d'exécution maximaux (pour garantir la synchronisation en temps voulu)

– ASTRÉE [BCC+03] est un analyseur statique qui vérifie l'absence d'erreurs à l'exécution

Résultats industriels obtenus avec ASTRÉE

Preuves automatiques d'absence d'erreur à l'exécution dans des logiciels de Commande De Vol Électrique :

- A340/600 : 132.000 lignes de C, 40mn sur un PC 2.8 GHz, 300 mégaoctets (nov. 2003)
- A380 : 1.000.000 de lignes de C, 34h, 8 gigaoctets (nov. 2005)

sans aucune fausse alarme

Premières mondiales !

3. Recherches et résultats actuels

Travaux théoriques

- Formalisation des descriptions du comportement des systèmes discrets complexes [6] et mécanisation des raisonnements sur ces systèmes en termes d'interprétation abstraite
- Propriétés numériques [7], symboliques [8] et de contrôle [9].

---

(6) traitement d'images [Ser94], systèmes biologiques [DFFK07, DFFK08, Fer07], calcul quantique [JP06], etc

(7) par exemple implantation efficace et correcte des polyèdres en flottants

(8) par exemple

- modèles mémoire de bas niveaux
- structures de données complexes
- protocoles cryptographiques

(9) par exemple le quasi-synchronisme, le parallélisme

Applications

- Nombreux prototypes d'analyseurs [10]
- Analyseurs de protocoles cryptographiques (propriétés de secret et de correspondance)
  - ProVerif (dans le modèle formel non borné, 2000–06)
  - CryptoVerif (dans le modèle calculatoire avec N sessions, 2005—)
- ASTRÉE (2001—)

[10] Par exemple :
- programmes orientés objet
- programmes quasi-synchrones
- tatoueur
- programmes mobiles
- analyses faiblement relationelles
- code compilé
- propriétés d'arbres

Transfert technologique

- Élargissement du champ d'application d'ASTRÉE (espace, moteurs d'avions,automobile, ferroviaire, télécommunications)
- Pérennisation du logiciel ASTRÉE (prototype $\rightarrow$ logiciel utilisable dans l'industrie)
  - A340 $\rightarrow$ à postériori
  - A380 $\rightarrow$ en accompagnement du développement
  - A350 $\rightarrow$ utilisation industrielle intégrée dans la chaîne de développement
- Certification d'ASTRÉE (pour l'industrie aéronautique)
- Industrialisation d'ASTRÉE (FIST, licences, Esterel technologie)

4. Projets et besoins futurs

Les grands challenges immédiats (2008-2011)

- Aide au diagnostique de l'origine des alarmes
- Propriétés de fatalité [11] pour les systèmes infinis
- Parallélisme

(11) liveness

Besoins à court terme

– Chercheurs stabilisés

2007 : Antoine MINÉ (CR2, CNRS), Xavier RIVAL (CR2, INRIA)

2008 : Jérôme FERET (CR2, INRIA), Laurent MAUBORGNE (délégation à l'INRIA Rocquencourt)

Départ : David MONNIAUX (CNRS) à VÉRIMAG

– **Ingénieur** pour aider au transfert technologique

– Thésards (ayant goût pour la théorie et le développement de logiciels)

5. Conclusion sur les enjeux stratégiques

Conclusion

- Vision : pour comprendre le monde numérique, il faut l'analyser à différents niveaux d'abstraction

- Théorie : l'interprétation abstraite assure une cohérence entre ces abstractions et offre des techniques d'approximation effectives pour les systèmes infinis

- Applications : le choix d'abstractions effectives suffisamment grossières pour être *calculables* et précises pour *éviter les fausses alarmes* permet de vaincre l'indécidabilité et la complexité dans la vérification automatique des modèles et des programmes

Les enjeux du futur

– Génie informatique → science de la programmation :
  - La croissance du logiciel n'est plus accompagnée par celle du matériel (parallélisme, compilation certifiée optimisante, méthodes de conception et vérification deviennent stratégiques)
  - La vérification manuelle par contrôle de la méthode de conception du programme sera complétée par la vérification automatique du programme produit
  - L'industrie du logiciel (embarqué) prend conscience de l'enjeu

– Stratégie de recherche :
  - Recherches “orphelines” (peu de possibilités de recrutement extérieur par permutation, essaimage unidirectionel)
  - Financements (à trop court terme, très compétitifs sur les promesses, peu favorables au développement)
  - Thèses (très courtes, peu favorables au développement)
  - Critères d’évaluation des équipes et des chercheurs (peu favorables au développement)
  - Coopérations industrielles (avec la recherche industrielle ou la production)
  - Administration de la recherche (ENS + CNRS + INRIA + ANR + AERES + Europe + Industriels)
  - Rémunération et reconnaissance (l’esprit de dévouement gratuit continuera-t-il longtemps?)

– Recherche :
  - Quelle part pour la recherche fondamentale (comment éviter la diminution?)
  - Quelle part pour la recherche appliquée (en forte augmentation)
  - Quelle part pour la l'administration de la recherche (en très forte augmentation!)
– Transfert industriel :
  - Quel modèle ?
  - Quel soutien ?
  - Quel marché ?
  - Quelle valorisation pour les chercheurs ?

– Pays émergents :
  - La Chine (et dans une moindre mesure l'Inde) ont compris l'enjeu de la qualité du logiciel
  - La concurrence va être terrible d'ici 5/10 ans

FIN

Merci de votre attention

6. Bibliography

Short bibliography

[AGM93] G. Amato, F. Giannotti, and G. Mainetto. Data sharing analysis for a database programming language via abstract interpretation. In R. Agrawal, S. Baker, and D.A.Bell, editors, *Proc. 19th Int. Conf. on Very Large Data Bases*, pages 405–415, Dublin, IE, 24–27 Aug. 1993. MORGANKAUFMANN.

[BCC+03] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. ACM SIGPLAN '2003 Conf. PLDI*, pages 196–207, San Diego, CA, US, 7–14 June 2003. ACM Press.

[Bla05] B. Blanchet. Security protocols: From linear to classical logic by abstract interpretation. *Inf. Process. Lett.*, 95(5):473–479, Sep. 2005.

[BPC01] J. Bailey, A. Poulovassilis, and C. Courtenage. Optimising active database rules by partial evaluation and abstract interpretation. In *Proc. 8th Int. Work. on Database Programming Languages*, LNCS 2397, pages 300–317, Frascati, IT, 8–10 Sep. 2001. Springer.

[BS97] V. Benzaken and X. Schaefer. Static integrity constraint management in object-oriented database programming languages via predicate transformers. In M. Aksit and S. Matsuoka, editors, *Proc. 11th European Conf. on Object-Oriented Programming, ECOOP '97*, LNCS 1241. Springer, Jyväskylä, FI, 9–13 June 1997.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL*, pages 238–252, Los Angeles, CA, 1977. ACM Press.

[CC79] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6th POPL*, pages 269–282, San Antonio, TX, 1979. ACM Press.

[CC92] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *19th POPL*, pages 83–94, Albuquerque, NM, US, 1992. ACM Press.

[CC95] P. Cousot and R. Cousot. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In *Proc. 7th FPCA*, pages 170–181, La Jolla, CA, US, 25–28 June 1995. ACM Press.

[CC00] P. Cousot and R. Cousot. Temporal abstract interpretation. In *27th POPL*, pages 12–25, Boston, MA, US, Jan. 2000. ACM Press.

[CC02] P. Cousot and R. Cousot. Systematic design of program transformation frameworks by abstract interpretation. In *29th POPL*, pages 178–190, Portland, OR, US, Jan. 2002. ACM Press.

[CC03] P. Cousot and R. Cousot. Parsing as abstract interpretation of grammar semantics. *Theoret. Comput. Sci.*, 290(1):531–544, Jan. 2003.

[CC04] P. Cousot and R. Cousot. An abstract interpretation-based framework for software watermarking. In *31st POPL*, pages 173–185, Venice, IT, 14–16 Jan. 2004. ACM Press.

[CCF+07] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Varieties of static analyzers: A comparison with ASTRÉE, invited paper. In M. Hinchey, He Jifeng, and J. Sanders, editors, *Proc. 1st TASE '07*, pages 3–17, Shanghai, CN, 6–8 June 2007. IEEE Comp. Soc. Press.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *5th POPL*, pages 84–97, Tucson, AZ, 1978. ACM Press.

[Cou97] P. Cousot. Types as abstract interpretations, invited paper. In *24th POPL*, pages 316–331, Paris, FR, Jan. 1997. ACM Press.

[Cou02] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoret. Comput. Sci.*, 277(1—2):47–103, 2002.

[Cou03] P. Cousot. Verification by abstract interpretation, invited chapter. In N. Dershowitz, editor, *Proc. Int. Symp. on Verification – Theory & Practice – Honoring Zohar Manna's 64th Birthday*, pages 243–268. LNCS 2772, Springer, Taormina, IT, 29 June – 4 Jul. 2003.

[DFF+07] V. Danos, J. Feret, W. Fontana, R. Harmer, and invited lecture J. Krivine. Rule-based modelling of cellular signalling. In L. Caires and V.-T. Vasconcelo, editors, *Proc. 18th Int. Conf. CONCUR '07*, number 4703 in LNCS, pages 17–41. Springer, Lisbon, PT, 3–8 Sep. 2007.

[DFFK07] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In Zhong Shao, editor, *Proc. 5th APLAS '2007*, pages 139–157, Singapore, 29 Nov. –1 Dec. 2007. LNCS 4807, Springer.

[DFFK08] V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract interpretation of cellular signalling networks. In F. Loggozzo, D. Peled, and L.D. Zuck, editors, *Proc. 9th Int. Conf. VMCAI 2008*, pages 83–97, San Francisco, CA, US, 7–9 Jan. 2008. LNCS 4905, Springer.

[DS07] D. Delmas and J. Souyris. ASTRÉE: from research to industry. In G. Filé and H. Riis-Nielson, editors, *Proc. 14th Int. Symp. SAS '07*, Kongens Lyngby, DK, LNCS 4634, pages 437–451. Springer, 22–24 Aug. 2007.

[Fer05a] J. Feret. The arithmetic-geometric progression abstract domain. In R. Cousot, editor, *Proc. 6th Int. Conf. VMCAI 2005*, pages 42–58, Paris, FR, 17–19 Jan. 2005. LNCS 3385, Springer.

[Fer05b] J. Feret. Numerical abstract domains for digital filters. In *1st Int. Work. on Numerical & Symbolic Abstract Domains, NSAD"05*, Maison Des Polytechniciens, Paris, FR, 21 Jan. 2005.

[Fer07] J. Feret. Reachability analysis of biological signalling pathways by abstract interpretation. In T.E. Simos and G. Maroulis, editors, *Computation in Modern Science and Engineering: Proc. 6th Int. Conf. on Computational Methods in Sciences and Engineering (ICCMSE'07)*, volume American Institute of Physics Conf. Proc. 963 (2, Part A & B), pages 619–622. AIP, Corfu, GR, 25–30 Sep. 2007.

[FHL+01] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. In T.A. Henzinger and C.M. Kirsch, editors, *Proc. 1st Int. Work. EMSOFT '2001*, volume 2211 of *LNCS*, pages 469–485. Springer, 2001.

[GM04] R. Giacobazzi and I. Mastroeni. Abstract non-interference: Parameterizing non-interference by abstract interpretation. In *31st POPL*, pages 186–197, Venice, IT, 2004. ACM Press.

[JP06] Ph. Jorrand and S. Perdrix. Towards a quantum calculus. In *Proc. 4th Int. Work. on Quantum Programming Languages, ENTCS*, 2006.

[Min06] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19:31–100, 2006.

[PCJD07] M. Dalla Preda, M. Christodorescu, S. Jha, and S. Debray. Semantics-based approach to malware detection. In *34th POPL*, pages 238–252, Nice, France, 17–19 Jan. 2007. ACM Press.

[Per06] S. Perdrix. *Modèles formels du calcul quantique : ressources, machines abstraites et calcul par mesure.* PhD thesis, Institut National Polytechnique de Grenoble, Laboratoire Leibniz, 2006.

[RT04] F. Ranzato and F. Tapparo. Strong preservation as completeness in abstract interpretation. In D. Schmidt, editor, *Proc. 30th ESOP '04*, volume 2986 of *LNCS*, pages 18–32, Barcelona, ES, Mar. 29 – Apr. 2 2004. Springer.

[RT06] F. Ranzato and F. Tapparo. Strong preservation of temporal fixpoint-based operators by abstract interpretation. In A.E. Emerson and K.S. Namjoshi, editors, *Proc. 7th Int. Conf. VMCAI 2006*, pages 332–347, Charleston, SC, US, 8–10 Jan. 2006. LNCS 3855 , Springer.

[Ser94] J. Serra. Morphological filtering: An overview. *Signal Processing*, 38:3–11, 1994.