AltaRica et CEGAR

Farès CHUCRI

Third AltaRica Workshop

27 Novembre 2007

Encadrement : Alain GRIFFAULT et Grégoire SUTRE

Outline

Les méthodes formelles

Objectifs

Concevoir et réaliser des systèmes sûrs.

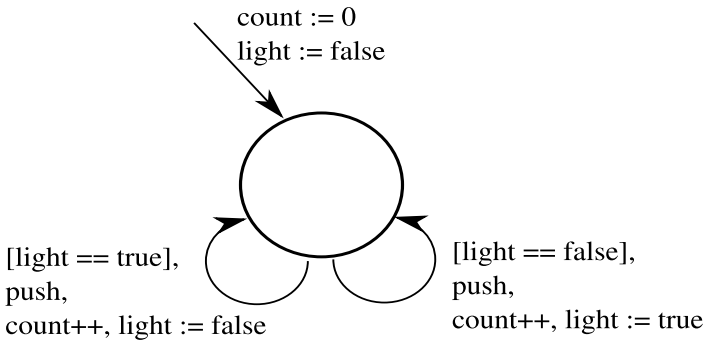
Les approches

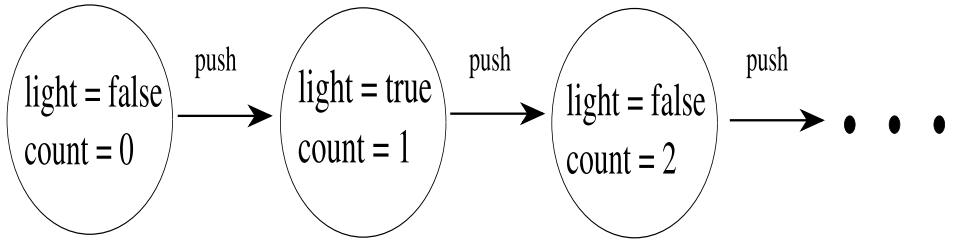
- Les démonstrateurs de théorèmes.
- Le raffinement (Méthode B).
- La génération automatique de code.
- La vérification de modèle.

La modélisation et les propriétés

AltaRica

Langage de description d'automates à contraintes
Sémantique : Un système de transition étiqueté.





Les propriétés

Logique du premier ordre(FO), logique temporisée(LTL),...
FO : $((count > 5) \wedge (\neg light))$

Vérification d'un modèle

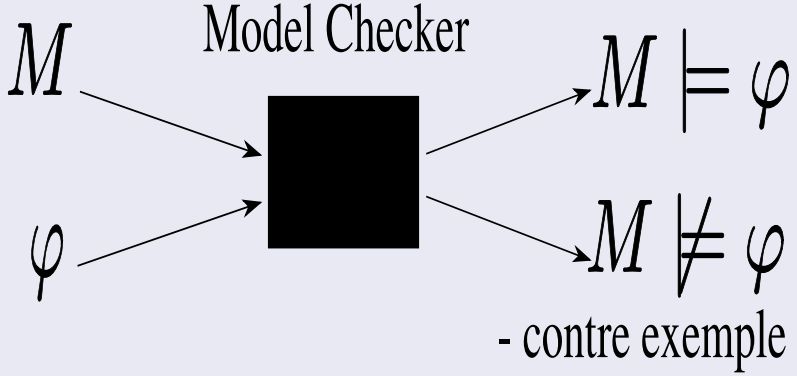
Modèle

Représentation symbolique d'un sytème.

Propriétés

- Sûreté
- Vivacité, et équité

Model-Checking

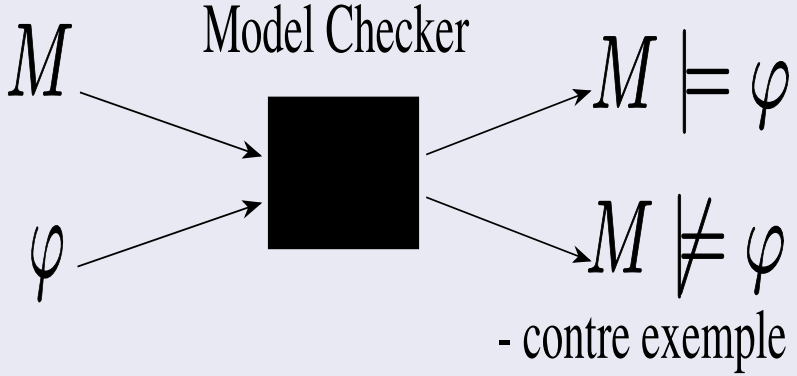


Vérification d'un modèle

Avantages/Inconvénient

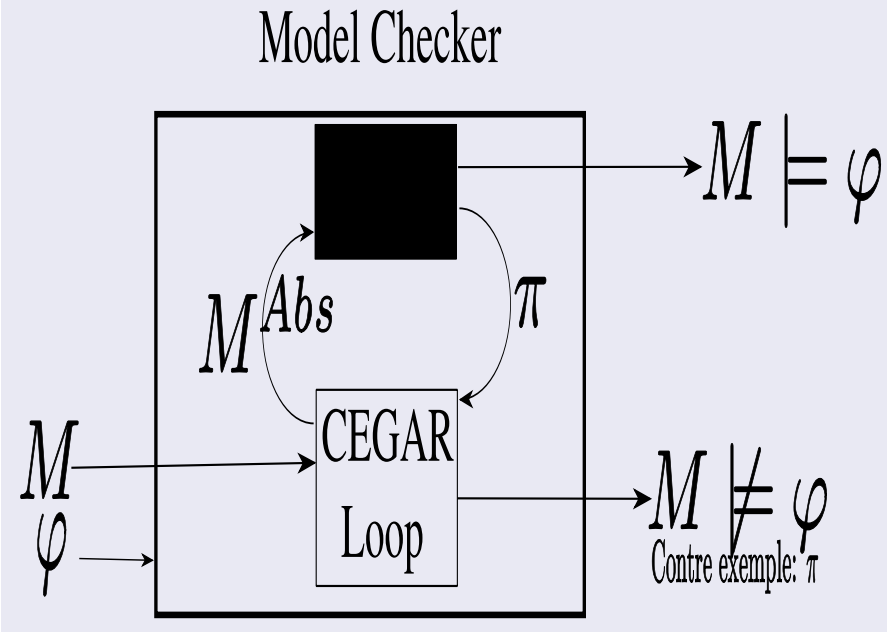
- Avantages
  - méthode sûre (exhaustive), contre-exemple.
  - Décidable pour les modèles finis.
- Inconvénients
  - Explosion combinatoire du nombre des états.
  - Indécidable pour les modèles infinis (semi-algorithmes).

Model-Checking

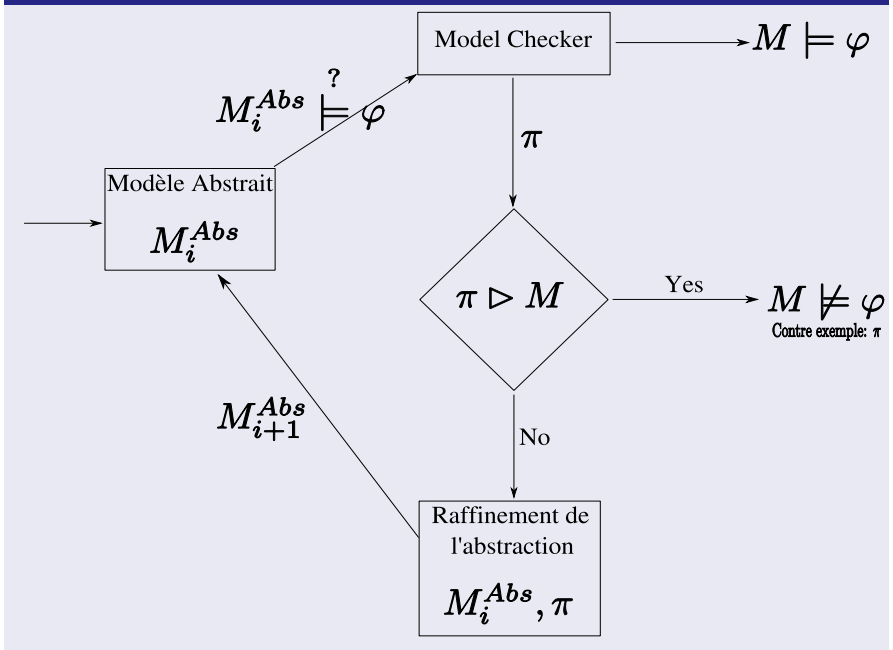


L'approche CEGAR : Counter Exemple Guided Abstraction Refinement [Clarcke et al.(2003)]

CEGAR Model Checking



CEGAR Loop



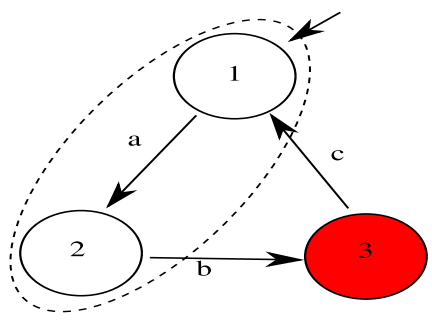
L'abstraction

L'abstraction

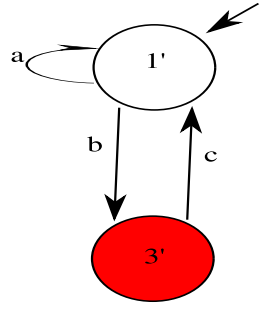
- Représentation approchée d'un modèle.
  - Abstraction des états.
  - Relation de transition abstraite.
- Simulation :
  - Tout comportement de M est un comportement de M^{Abs}.

Avantage/Inconvénient

- $|M^{Abs}| \leq |M|$.
- Contre-exemples non réalisables.



M



M^{Abs}

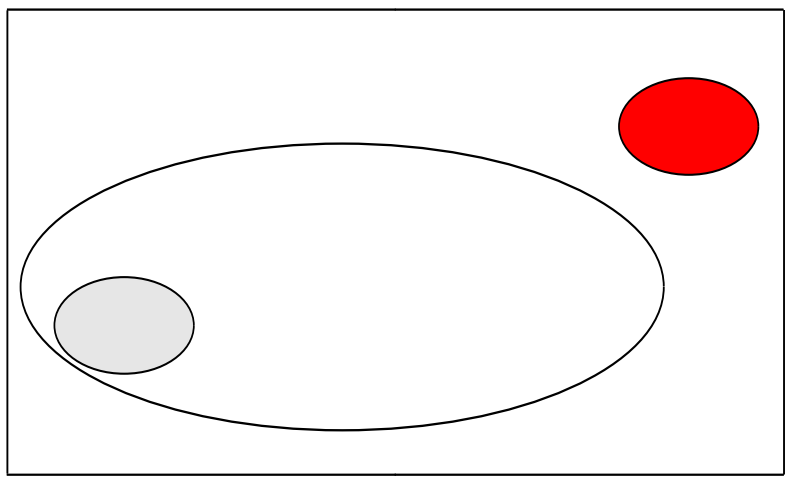
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



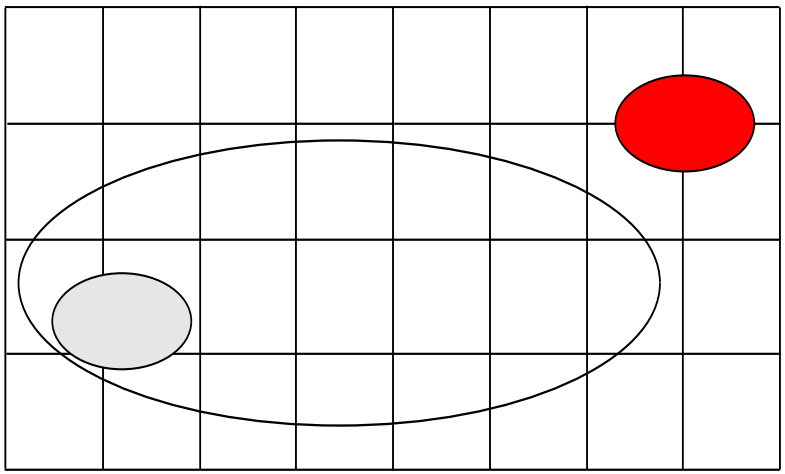
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



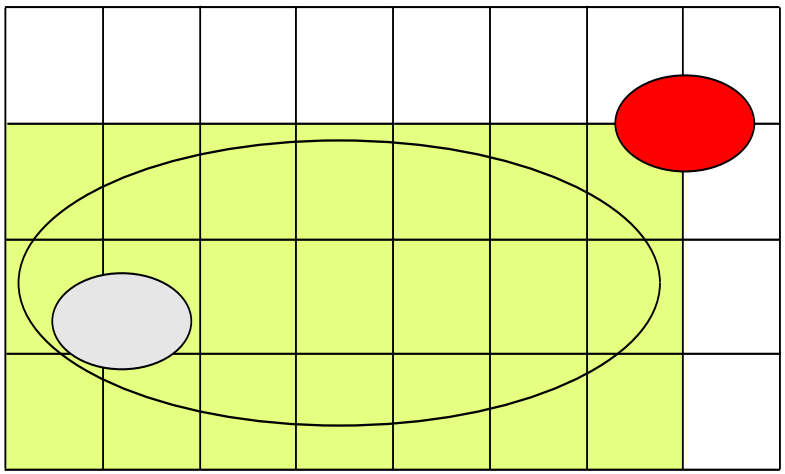
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



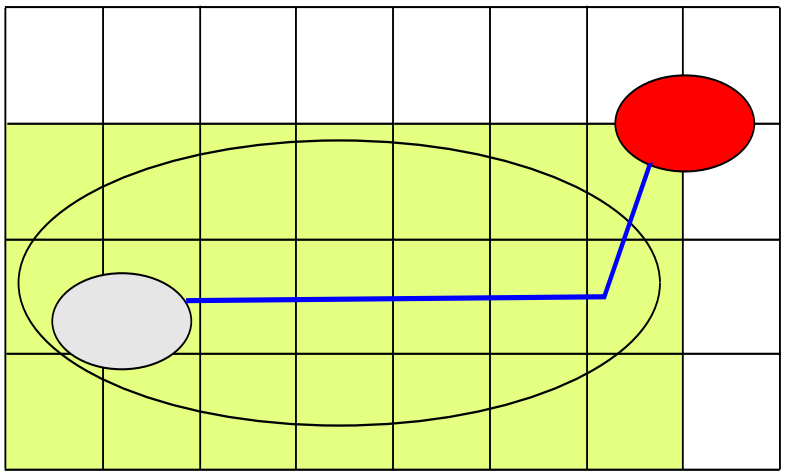
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



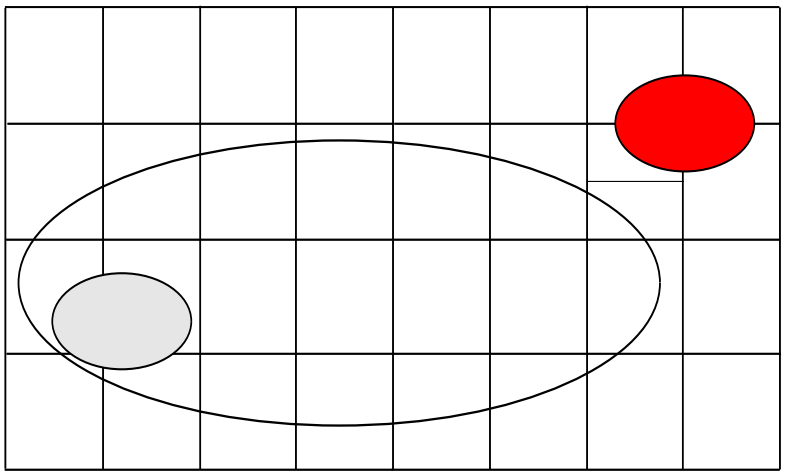
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



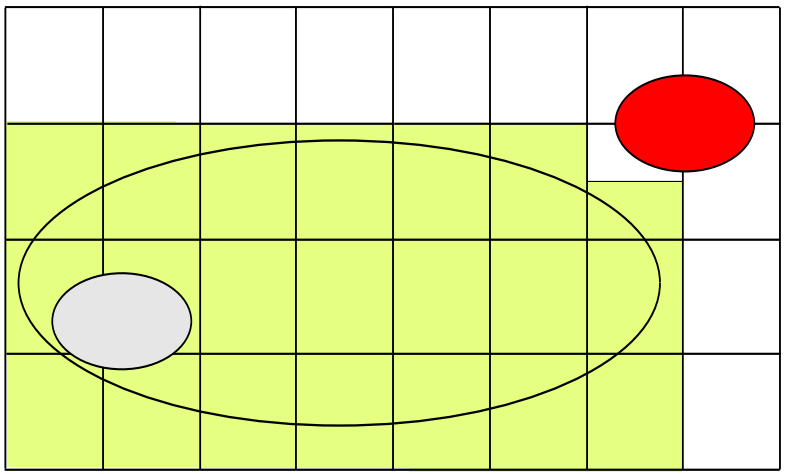
Un exemple : l'abstraction par prédicats [Graf and Saidi(1997)]

Principe

Utiliser des prédicats $P_1, \ldots, P_n$ pour partitionner l'espace des états.

Exemple

1. Calculer la partition.
2. Calculer les accessibles.
3. Vérifier la propriété.
4. Raffiner l'abstraction.
5. Calculer les accessibles.



Conservation des propriétés de sûreté

Les propriétés de sûreté

- Configuration ou comportement que l'on ne désire jamais rencontrer.
- Contre-exemple : chemin fini dans la modélisation.

Théorème

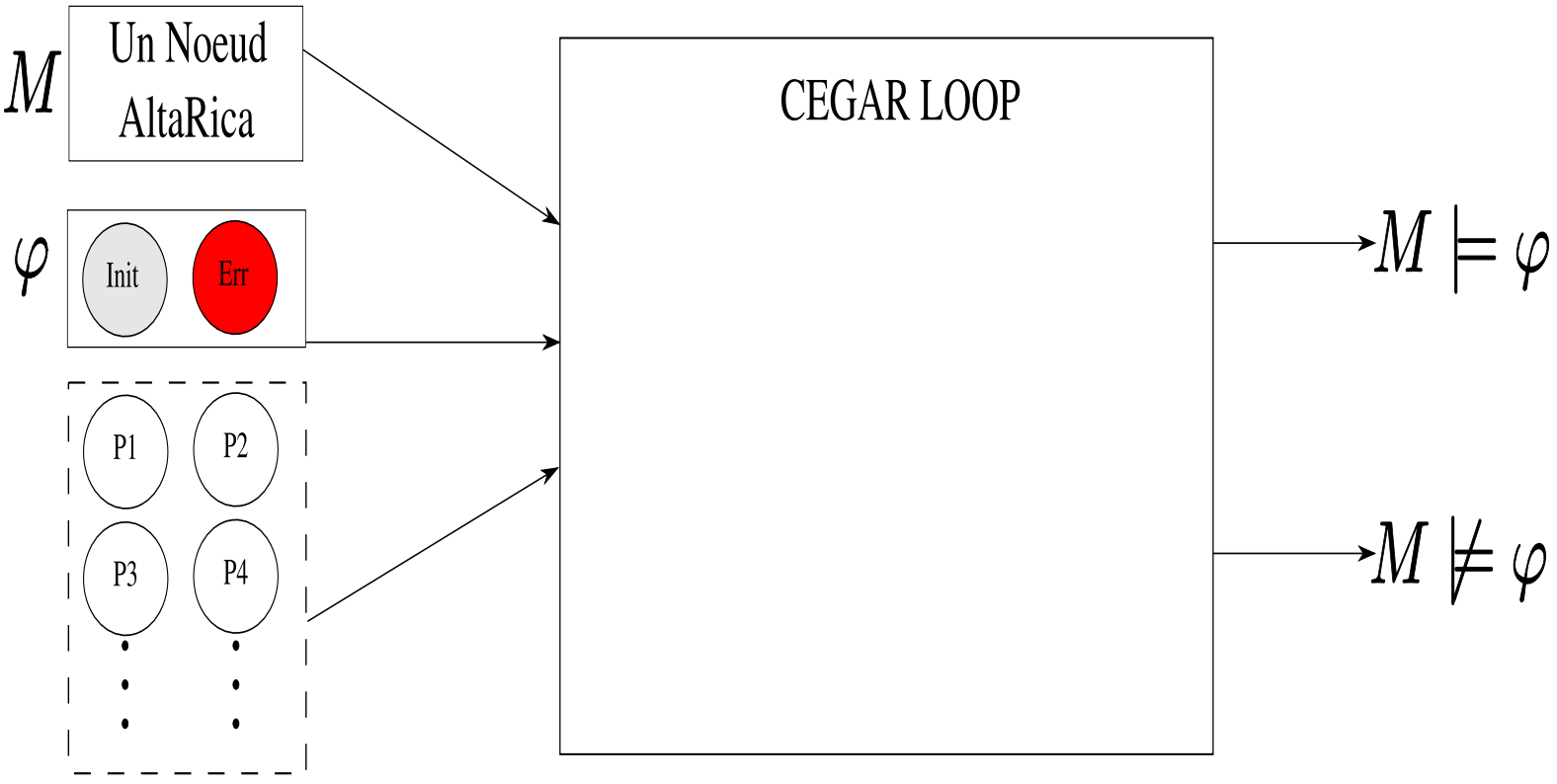
Soit M^{Abs} une abstraction d'un modèle M, φ une propriété de sûreté. Alors $M^{Abs} \models \varphi \Rightarrow M \models \varphi$.

Outline

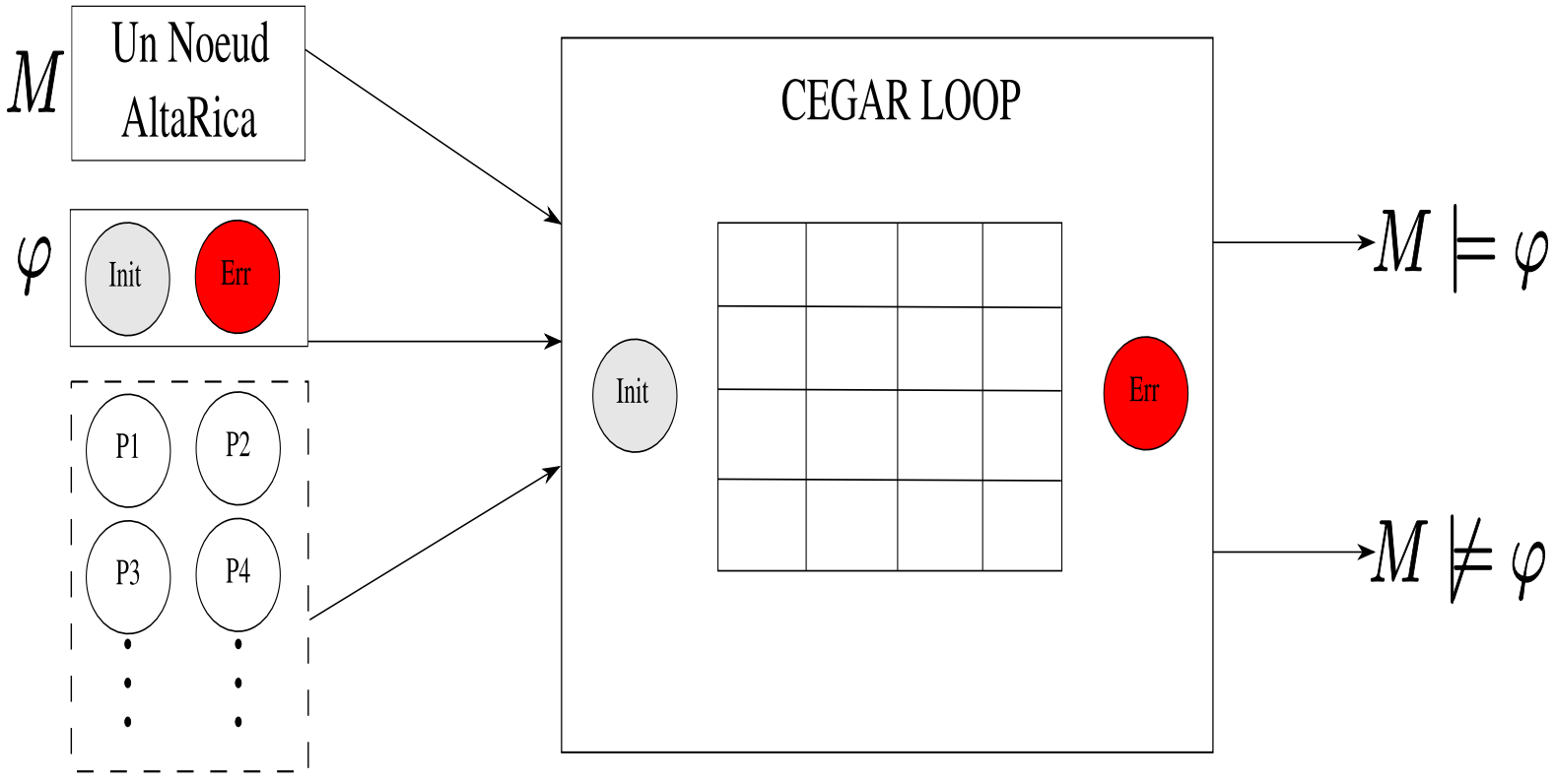
Algorithme CEGAR [Clarke et al.(2003)Clarke, Grumberg, Jha, Lu, and Veith]

1. Calculer une abstraction M^{Abs} de M.
2. Vérifier $M^{Abs} \models \varphi$
3. Si la propriété est satisfaite alors **retourner** $M \models \varphi$
4. Sinon vérifier la trace abstraite π
   - Si la trace abstraite est vérifiée **retourner** $M \not\models \varphi$ et π
   - Sinon Raffiner l'abstraction, et recommencer à l'étape 2

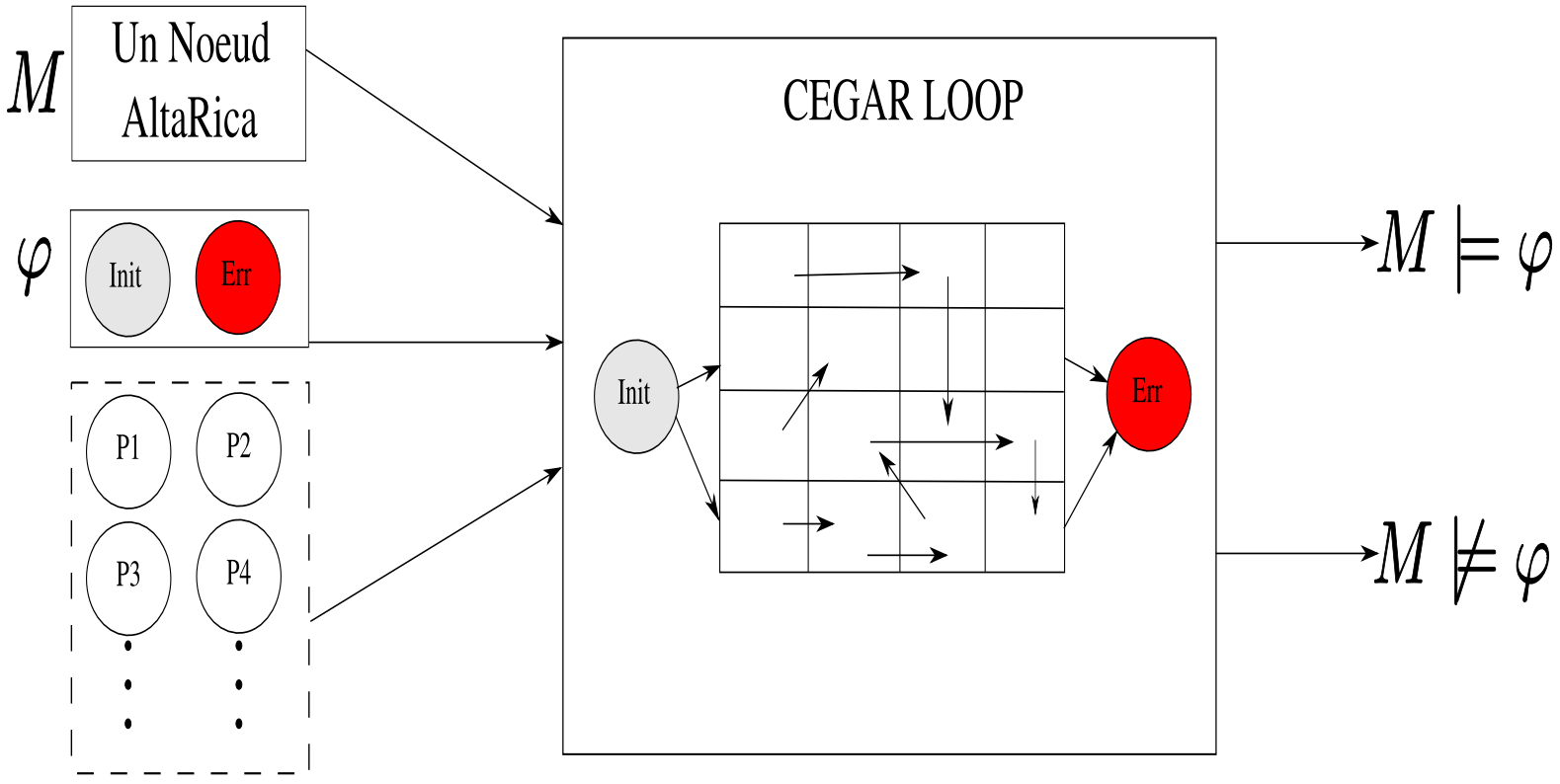
L'abstraction et le raffinement



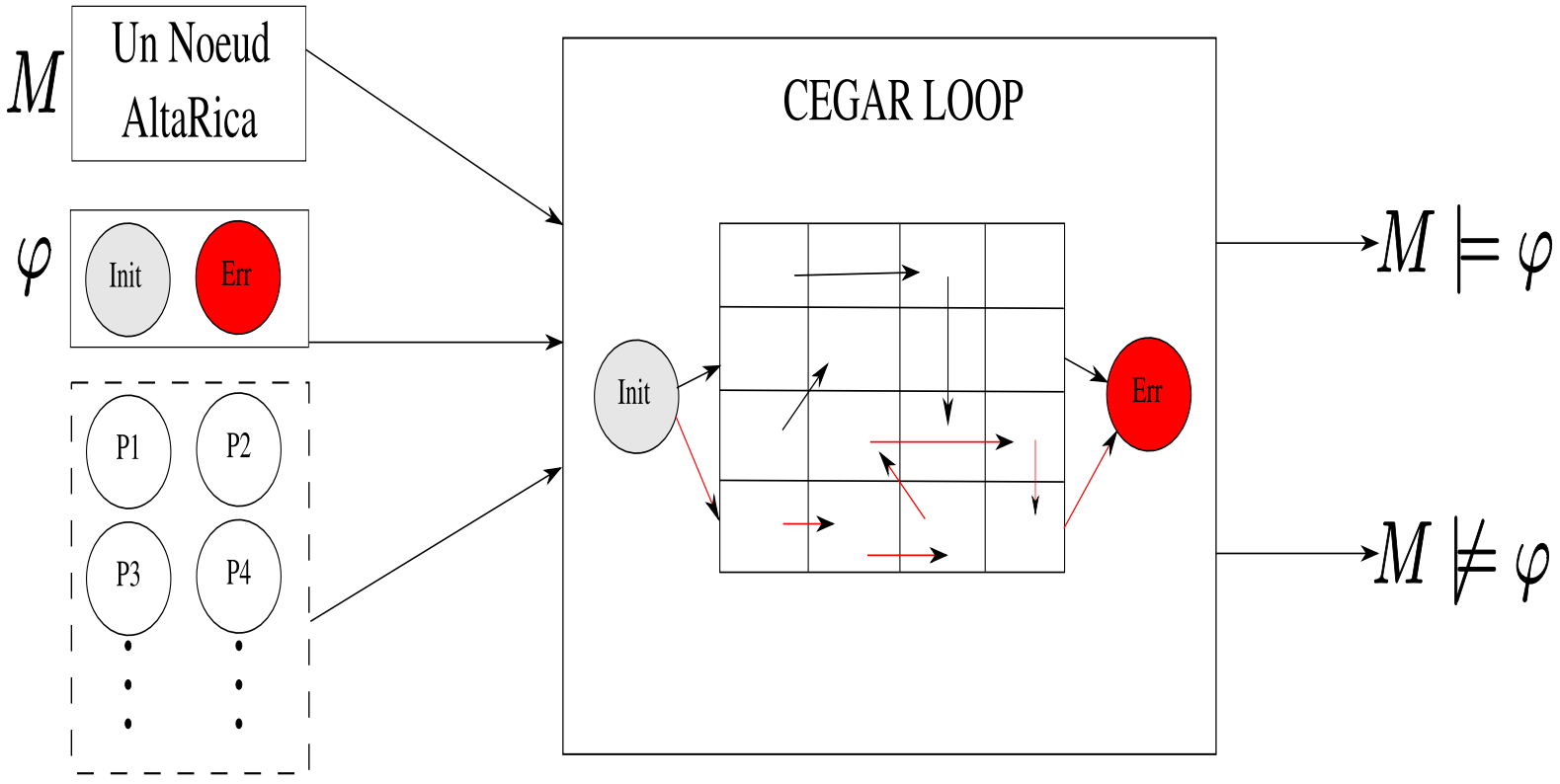
L'abstraction et le raffinement



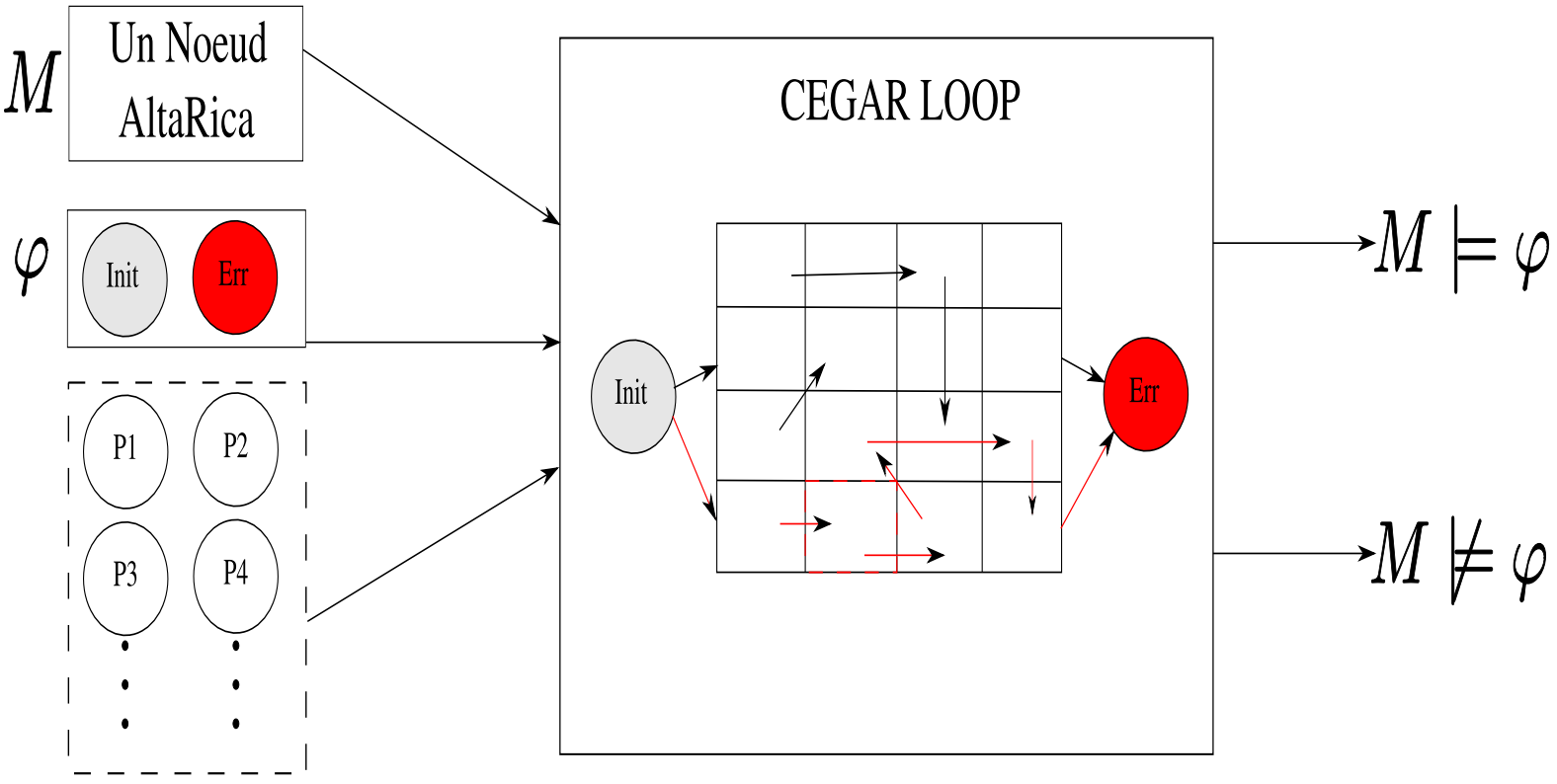
L'abstraction et le raffinement



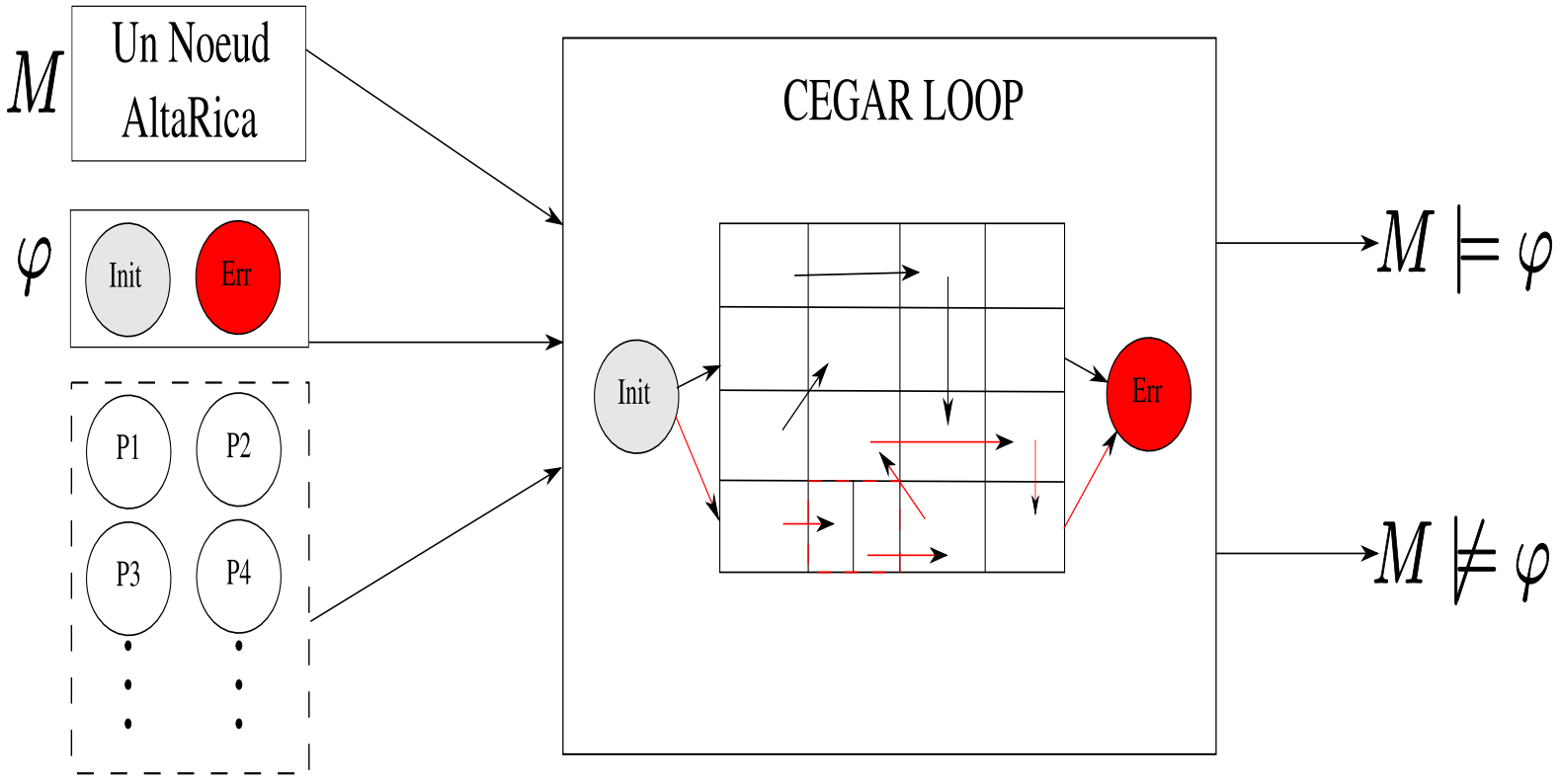
L'abstraction et le raffinement



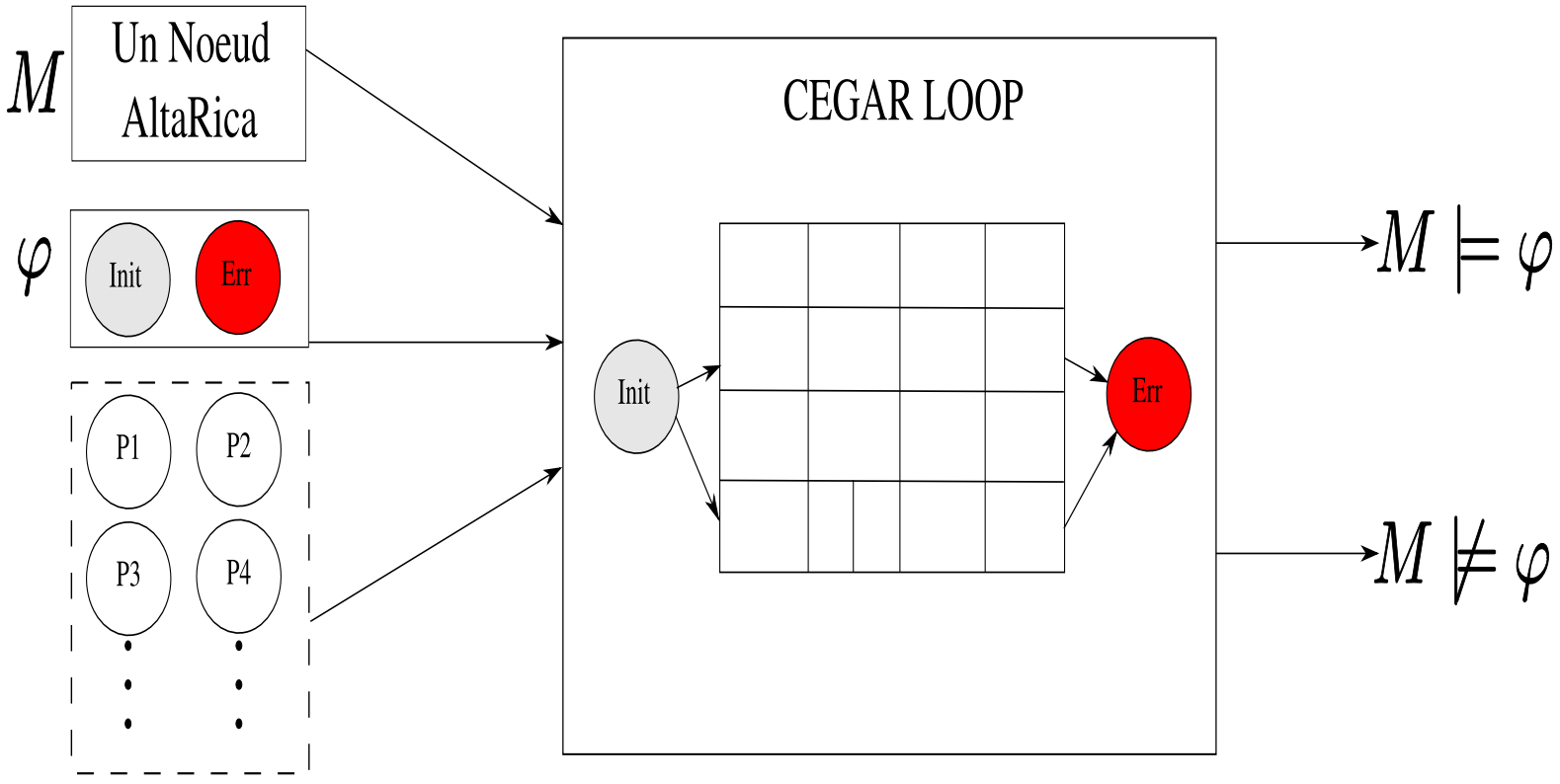
L'abstraction et le raffinement



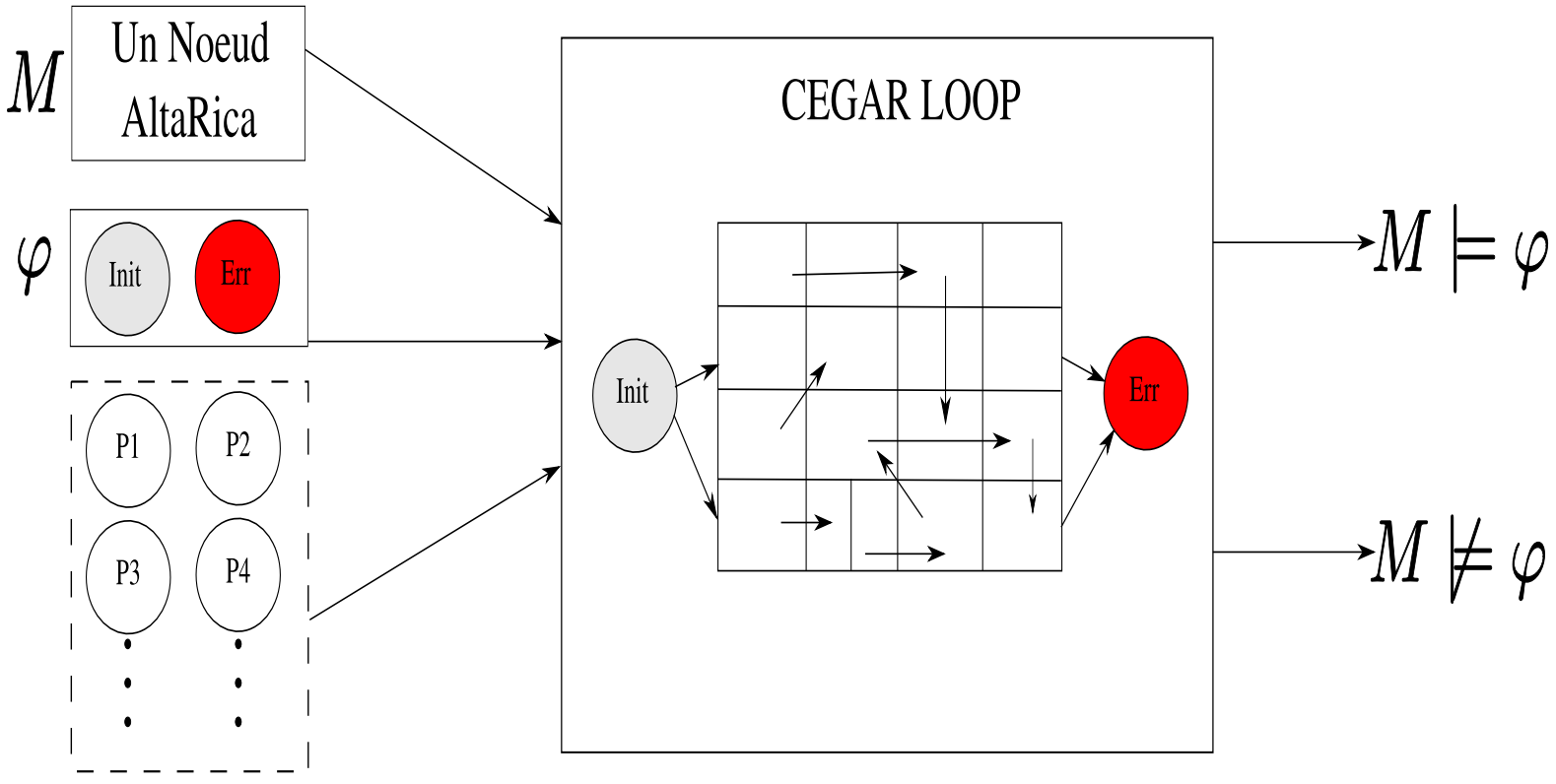
L'abstraction et le raffinement



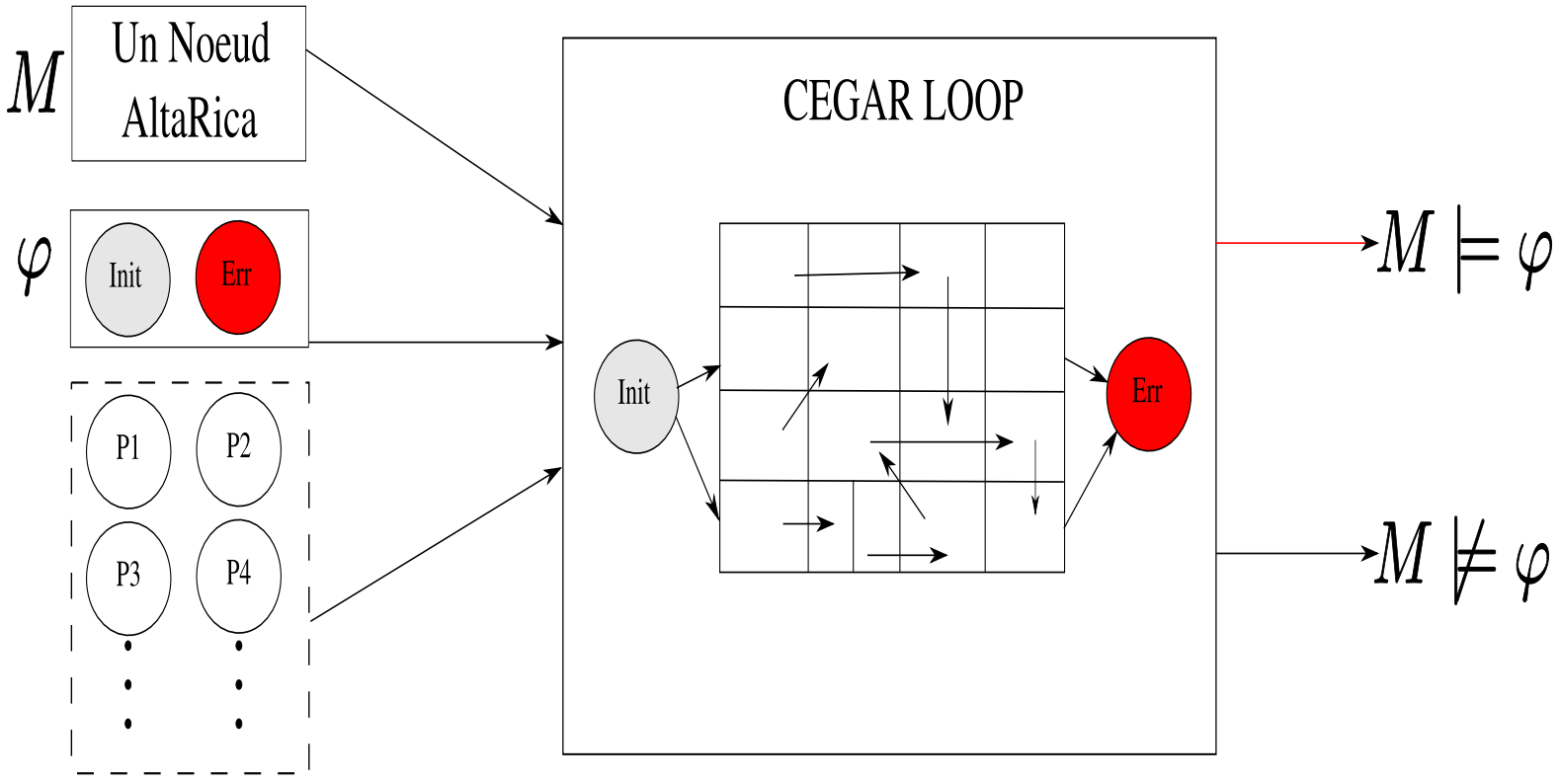
L'abstraction et le raffinement



L'abstraction et le raffinement



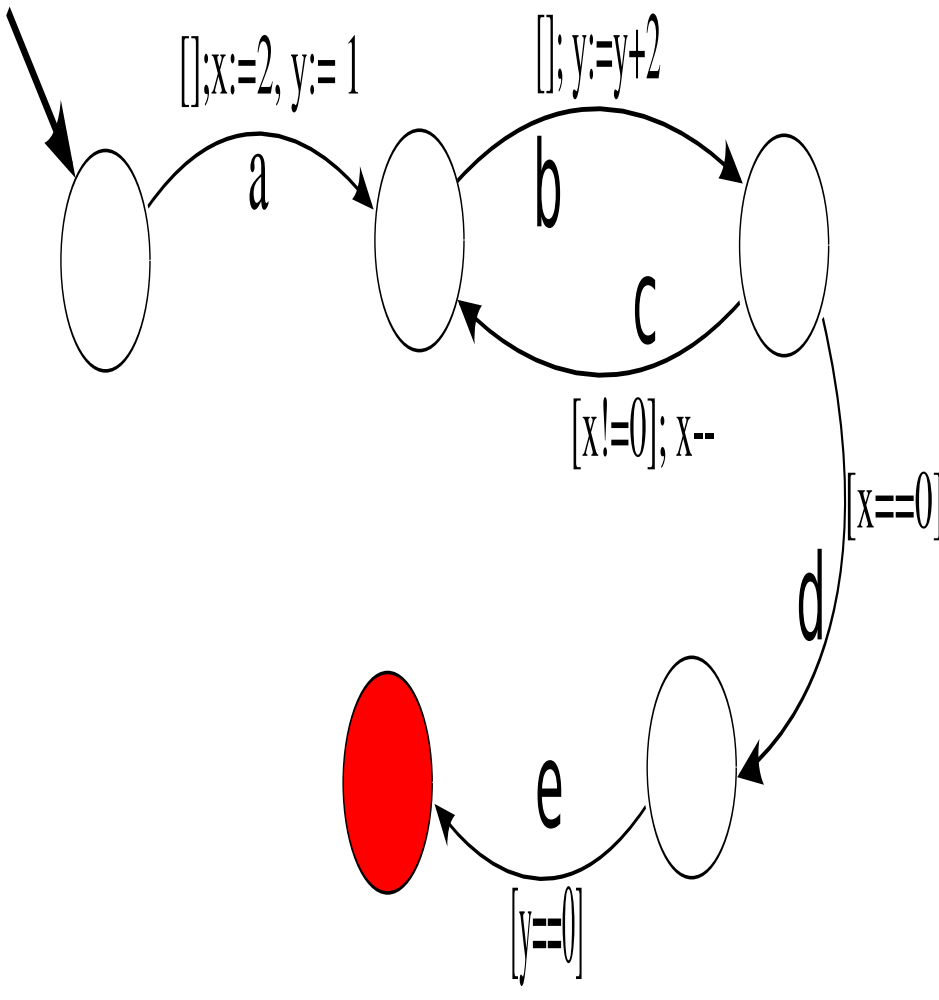
L'abstraction et le raffinement



Un exemple : le nœud foo

Un nœud AltaRica

```
node foo
  state
   x : [0,10] : public;
   y : [0,10] : public;
   count : [0,4] : public;
  event
   a,b,c,d,e;
  trans
   count = 0 |- a -> x := 2, y := 1, count := 1;
   count = 1 |- b -> y := y + 2, count := 2;
   count = 2 |- c -> x := x - 1, count := 1;
   count = 2 & x = 0 |- d -> count := 3;
   count = 3 & y = 0 |- e -> count := 4;
edon
```

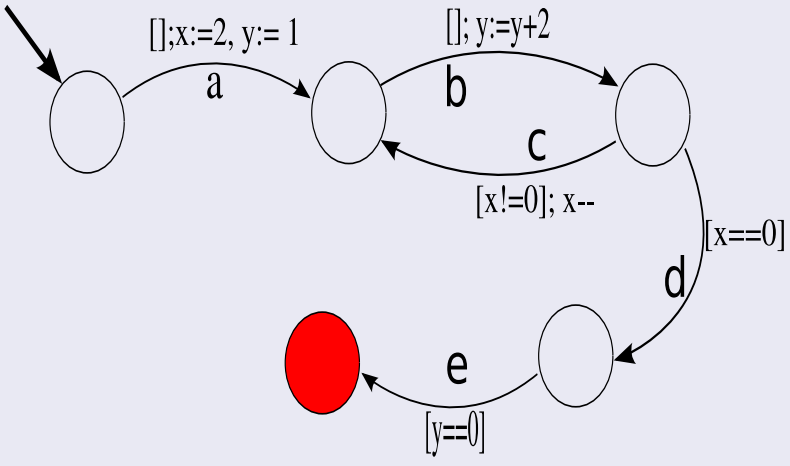


La vérification du nœud foo par l'approche CEGAR

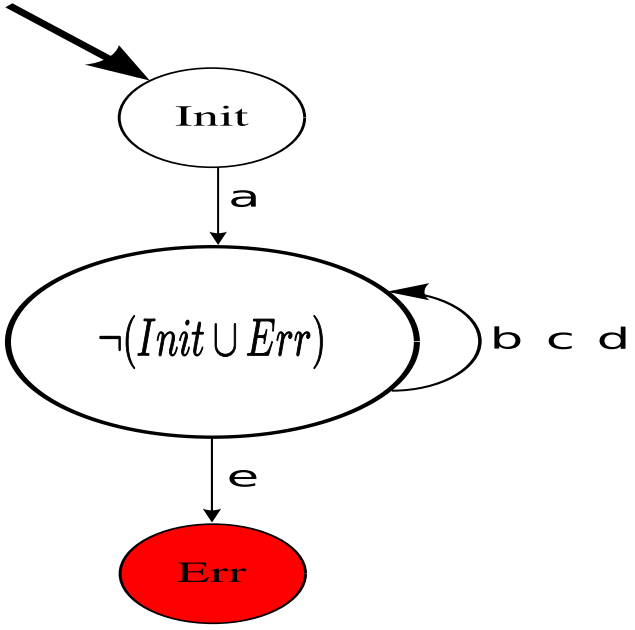
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



L'abstraction initiale

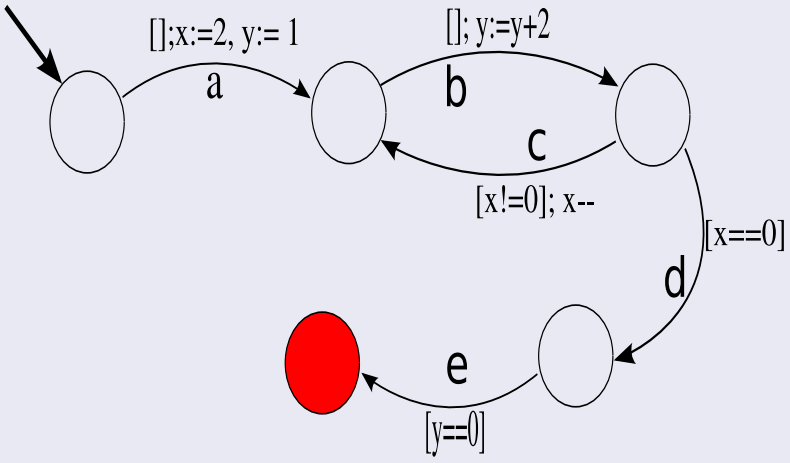


La vérification du nœud foo par l'approche CEGAR

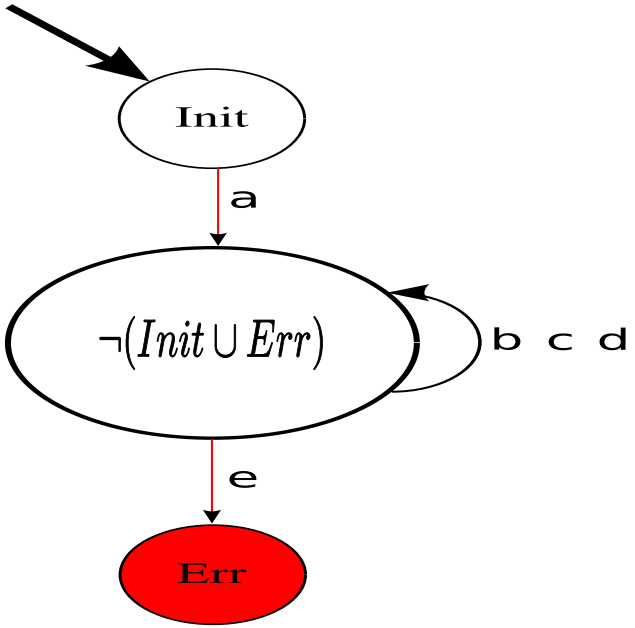
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Vérification de la propriété

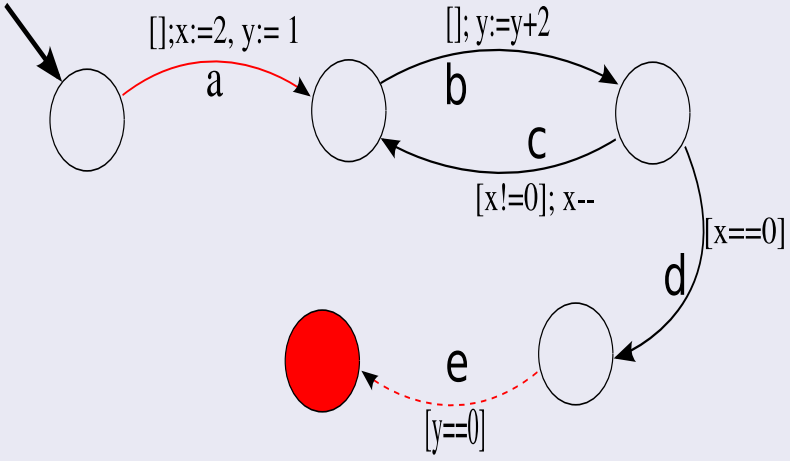


La vérification du nœud foo par l'approche CEGAR

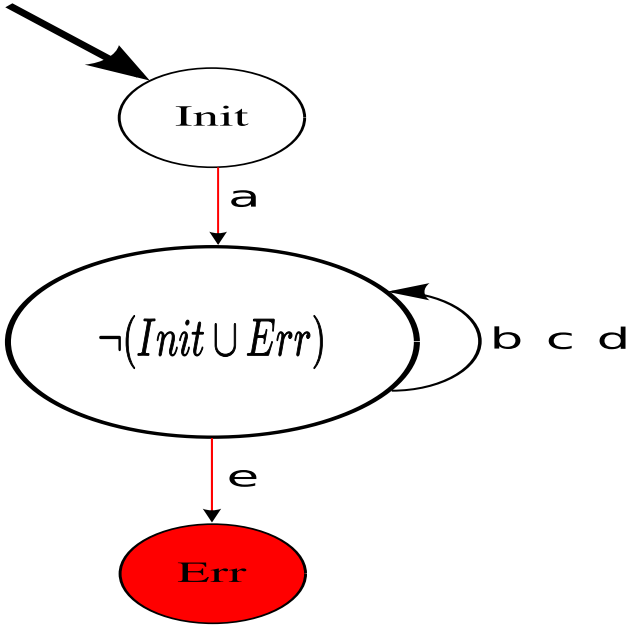
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Validation du contre-exemple

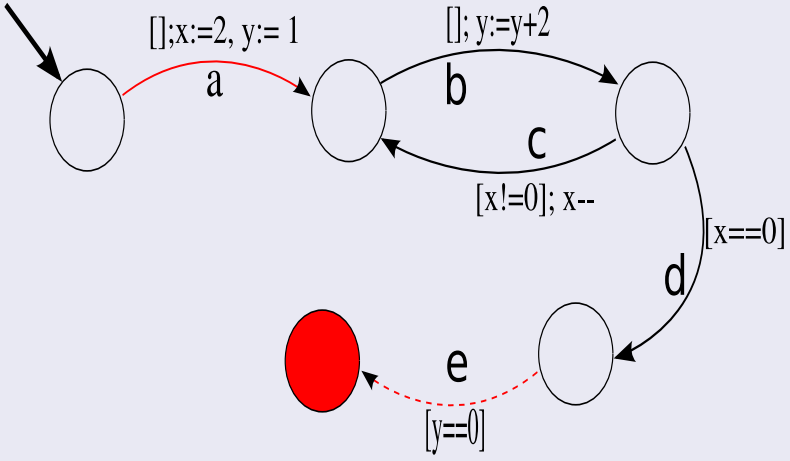


La vérification du nœud foo par l'approche CEGAR

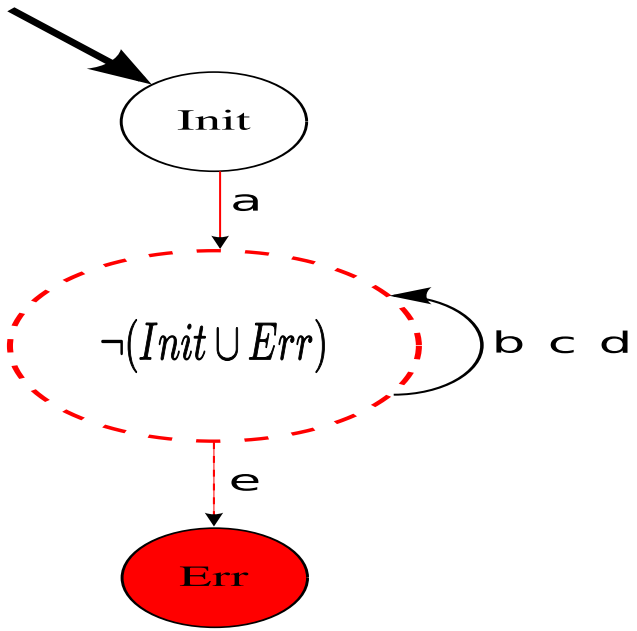
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Recherche de l'état à partager

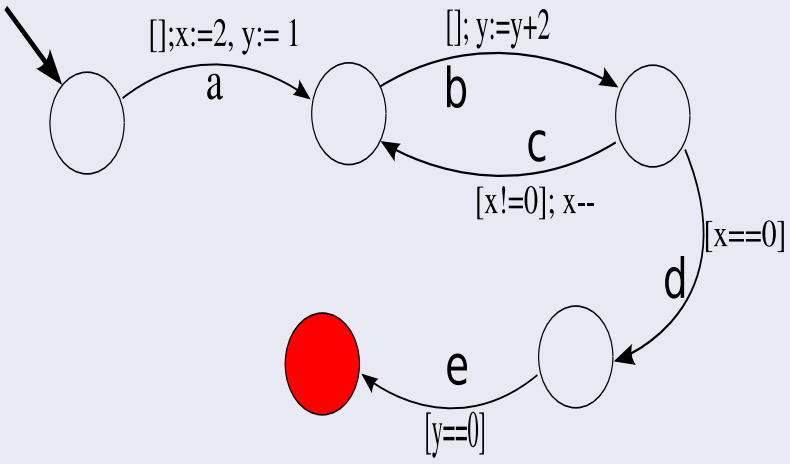


La vérification du nœud foo par l'approche CEGAR

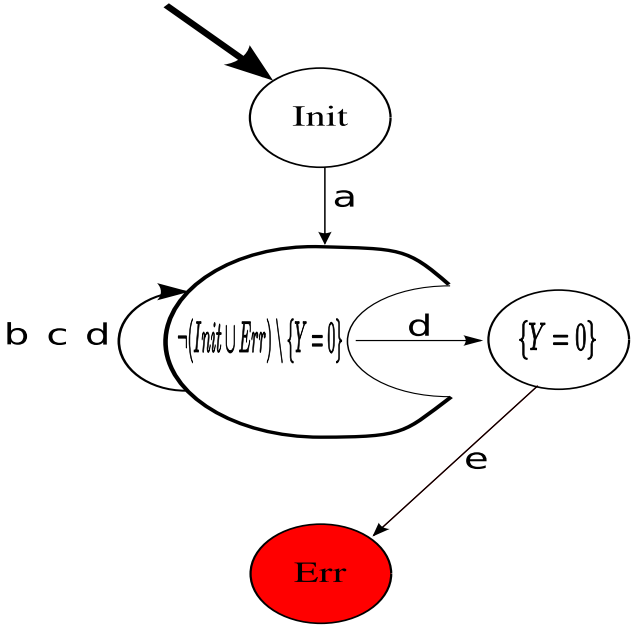
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Raffinement du modèle abstrait

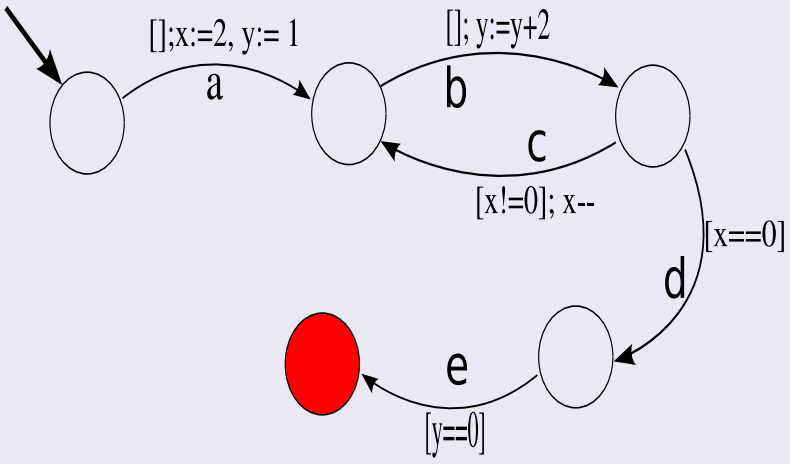


La vérification du nœud foo par l'approche CEGAR

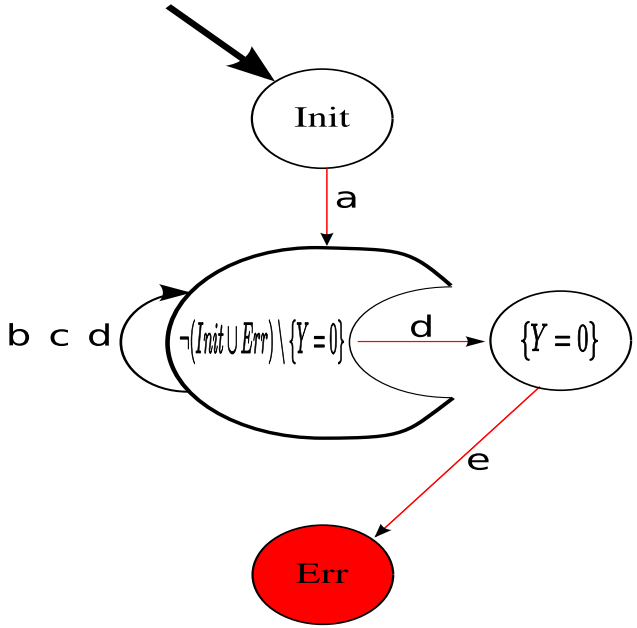
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Vérification de la propriété

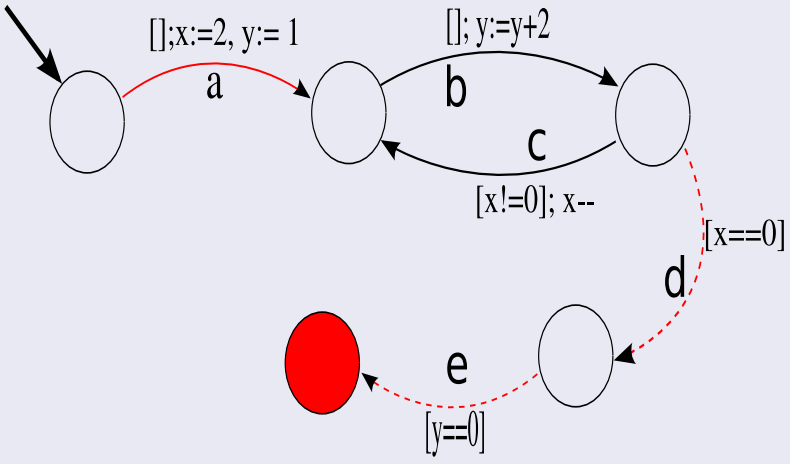


La vérification du nœud foo par l'approche CEGAR

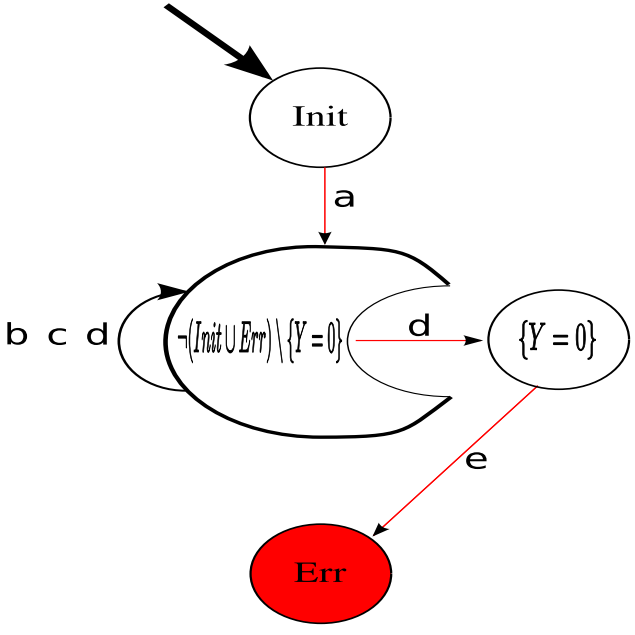
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Validation du contre-exemple

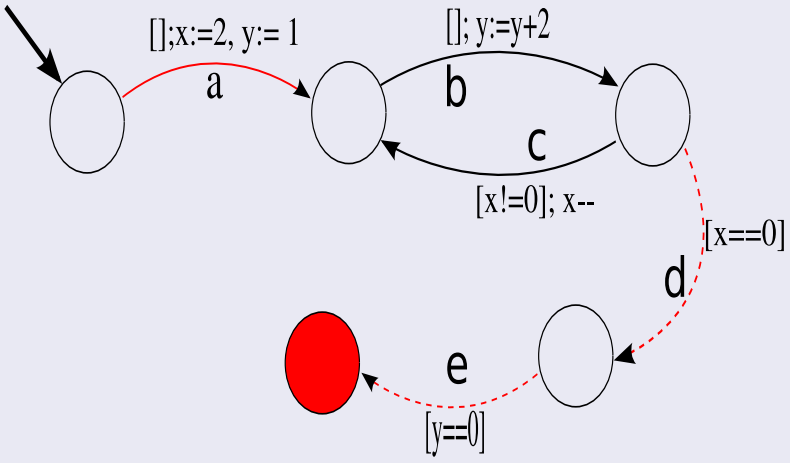


La vérification du nœud foo par l'approche CEGAR

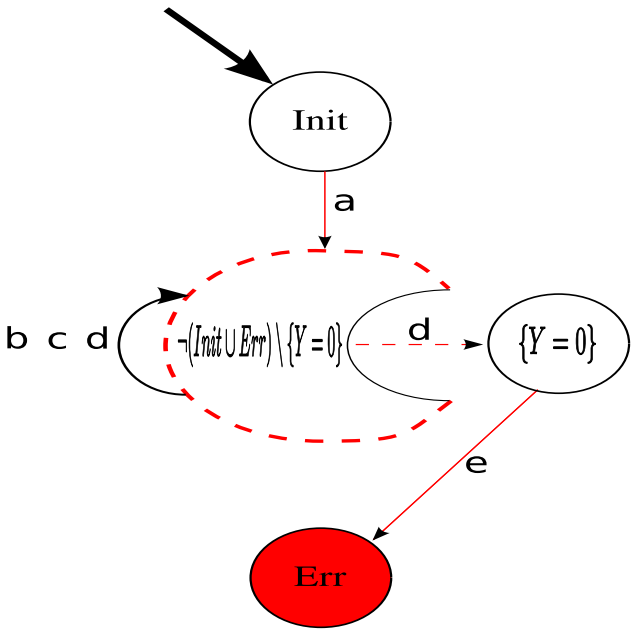
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Recherche de l'état à partager

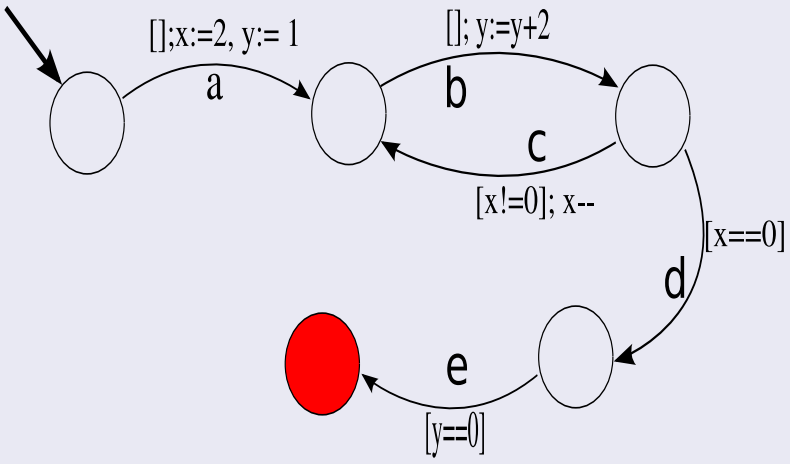


La vérification du nœud foo par l'approche CEGAR

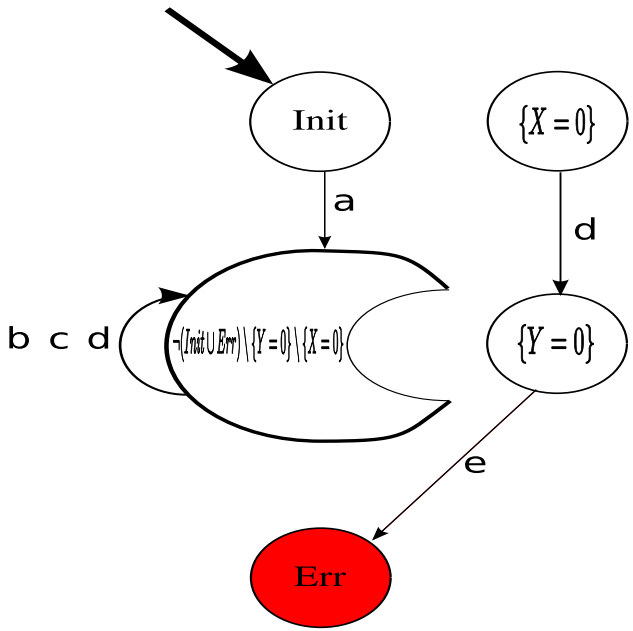
Propriété

On n'atteint jamais une configuration où count = 4.

Le modèle concret



Vérification de la propriété



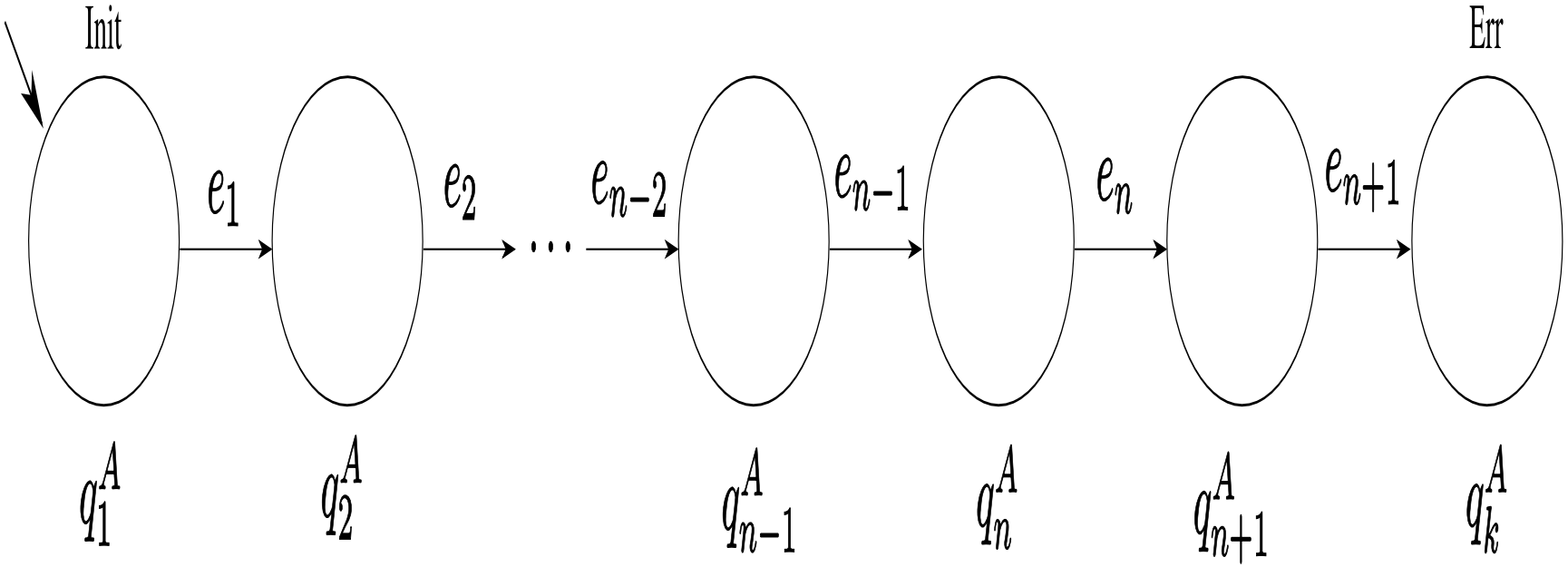
Vérification du contre-exemple

Objectif

Reproduire le contre-exemple dans le modèle concret.

Propriété

$Post(e_i) \cap Pre(e_{i+1}) \neq \emptyset$



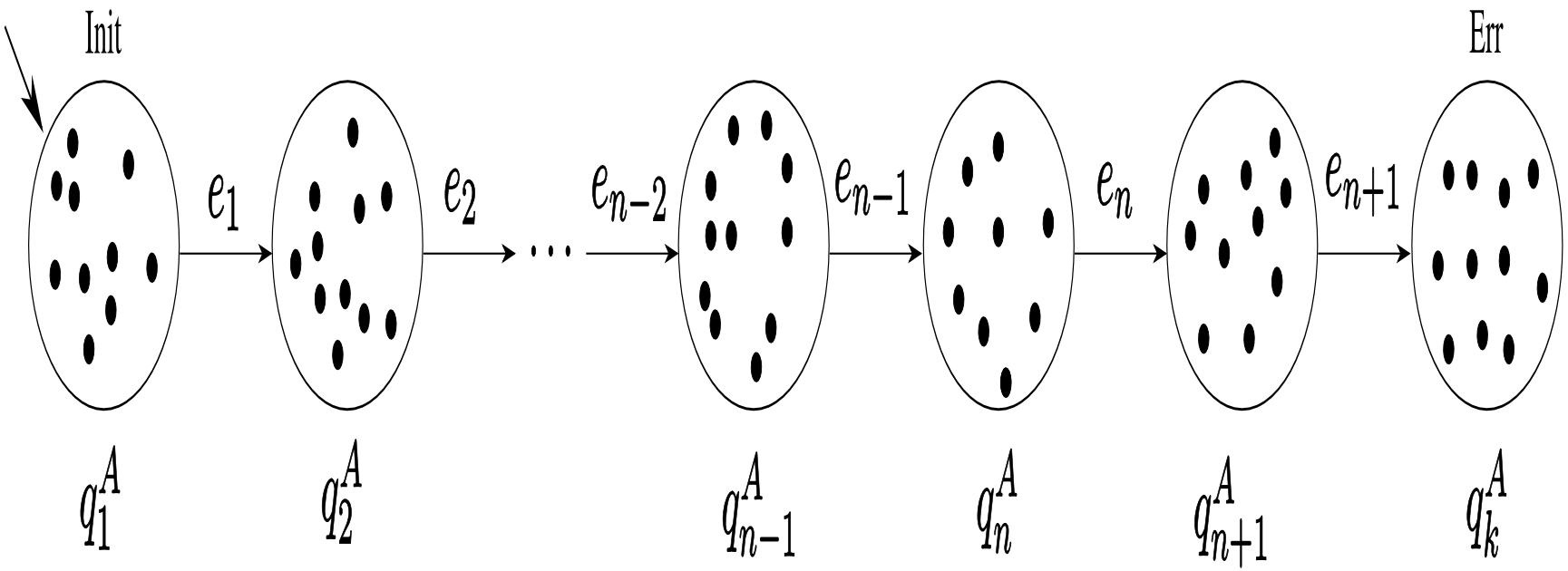
Vérification du contre-exemple

Objectif

Reproduire le contre-exemple dans le modèle concret.

Propriété

$Post(e_i) \cap Pre(e_{i+1}) \neq \emptyset$



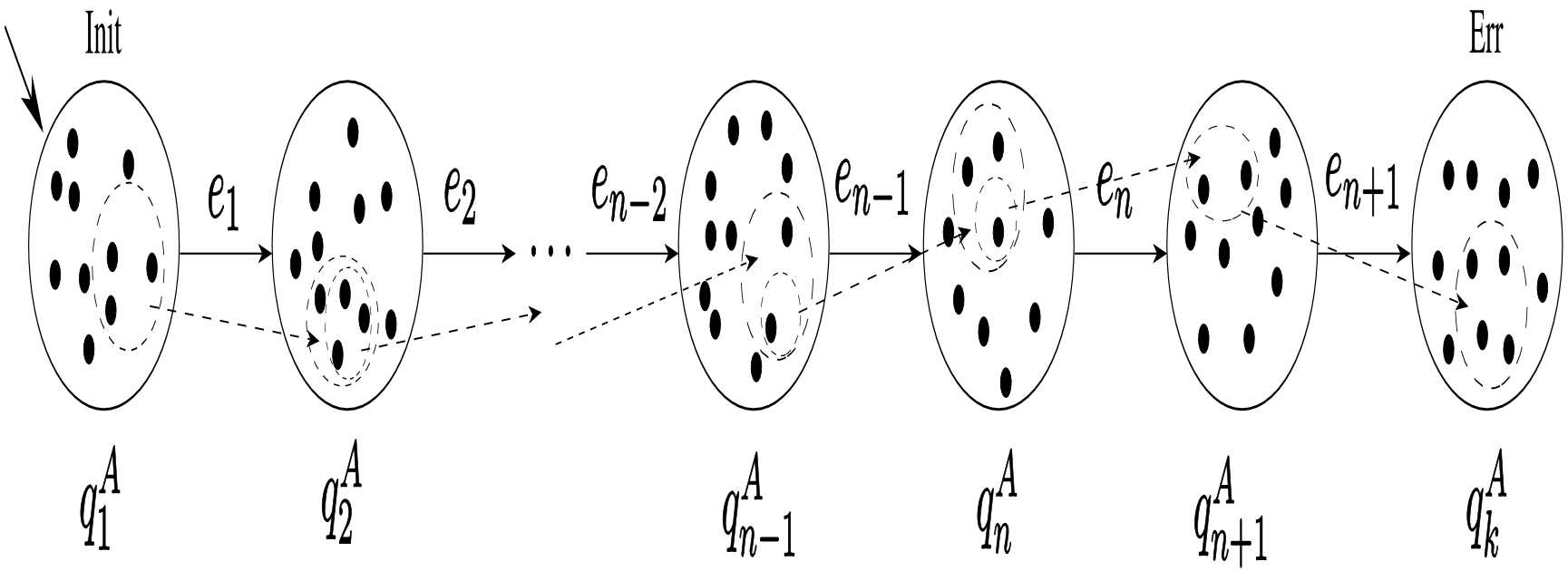
Vérification du contre-exemple

Objectif

Reproduire le contre-exemple dans le modèle concret.

Propriété

$Post(e_i) \cap Pre(e_{i+1}) \neq \emptyset$



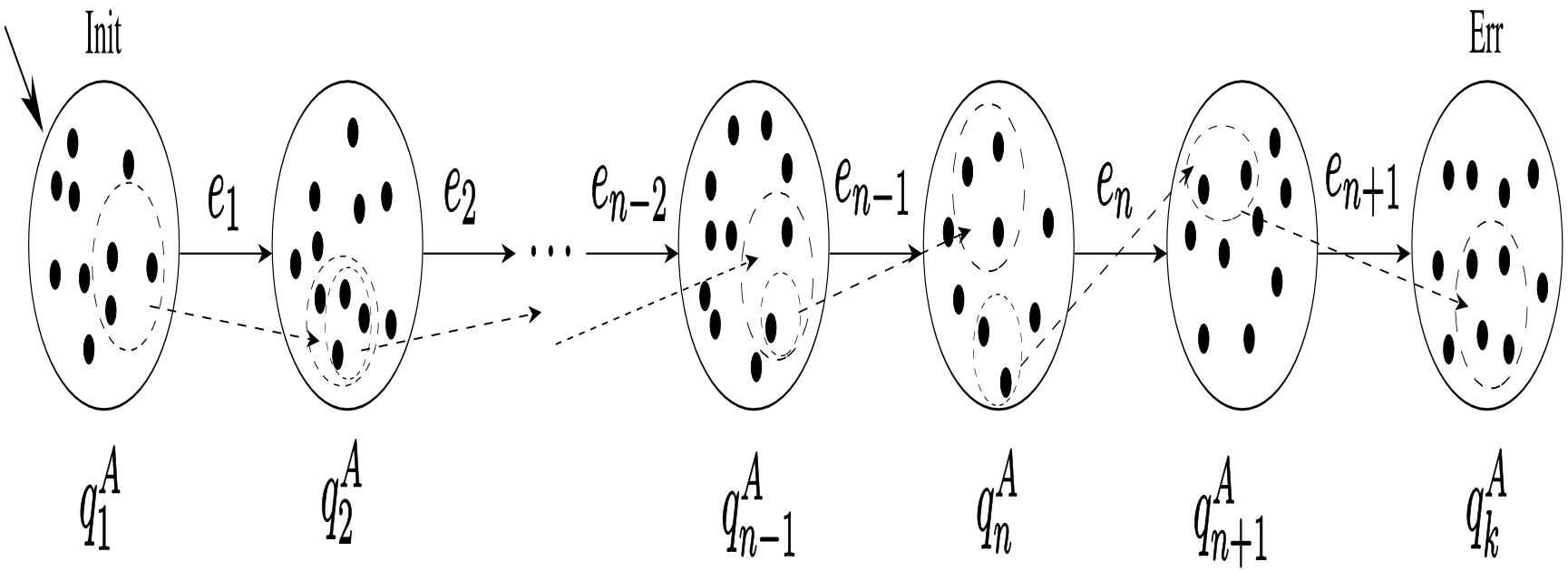
Vérification du contre-exemple

Objectif

Reproduire le contre-exemple dans le modèle concret.

Propriété

$Post(e_i) \cap Pre(e_{i+1}) \neq \emptyset$



Outline

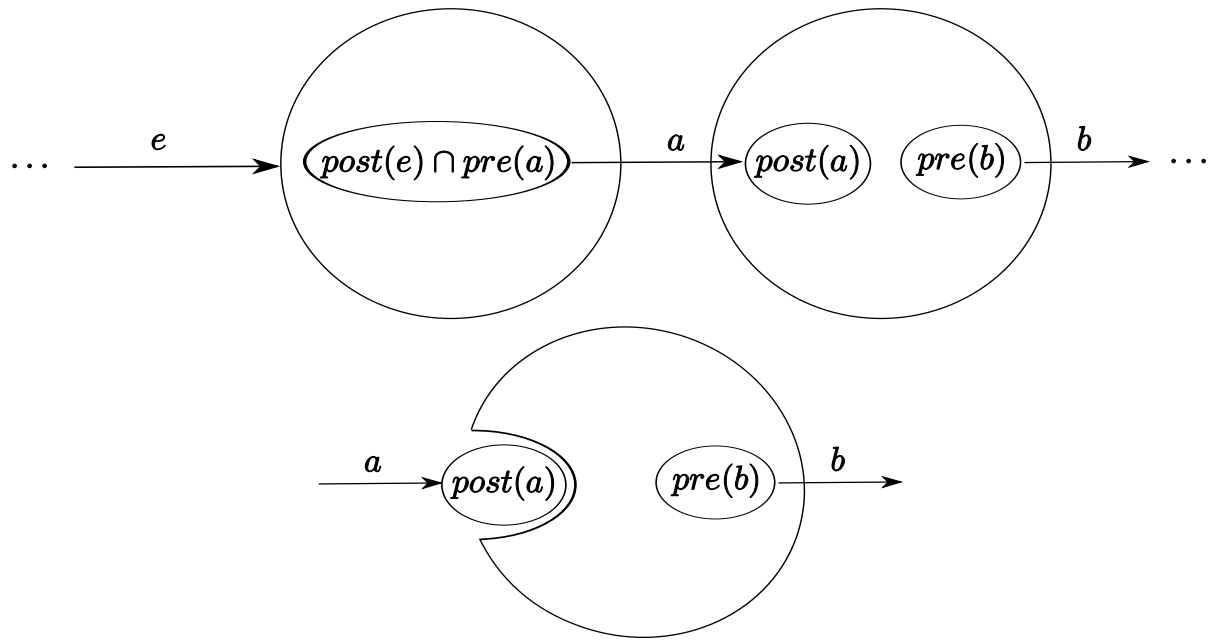
Travaux en cours

1. Etude des techniques de séparation
2. Recalcul de la relation de transition abstraite
3. Benchmarks

Heuristique 1
[Clarke et al.(2003)Clarke, Grumberg, Jha, Lu, and Veith]

Principe

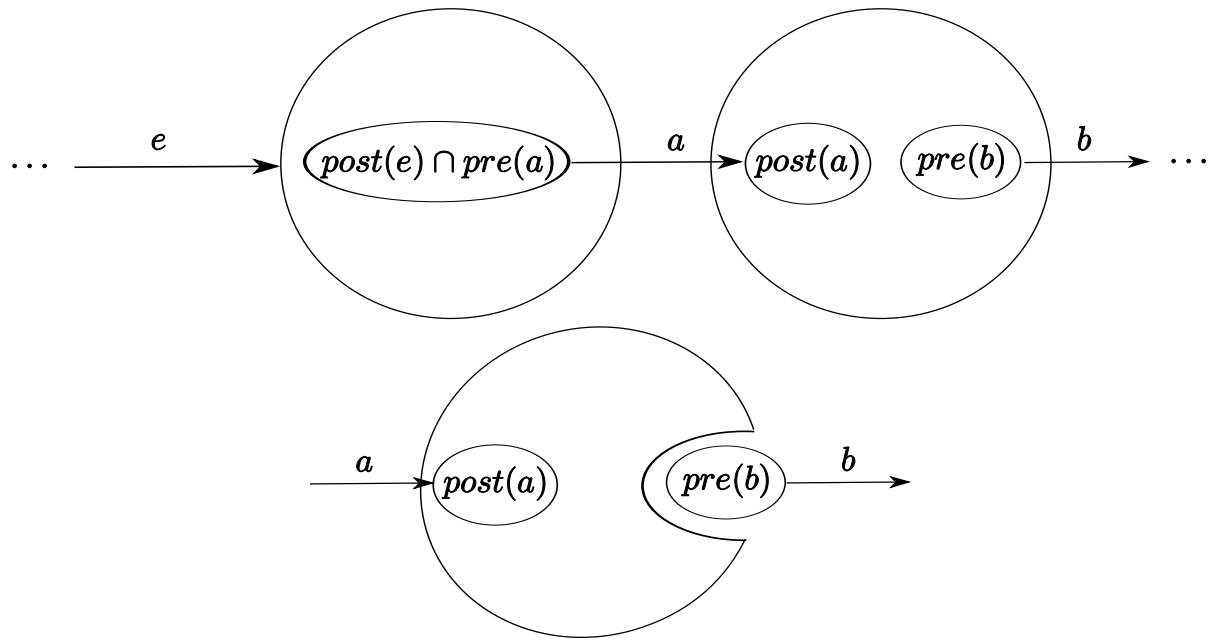
Utiliser le dernier sous-ensemble d'états concrets accessible à partir de l'état initial.



Heuristique 2 (Nouvelle variante)

Principe

Utiliser l'ensemble de tous les prédécesseurs concrets du premier nœud inaccessible.



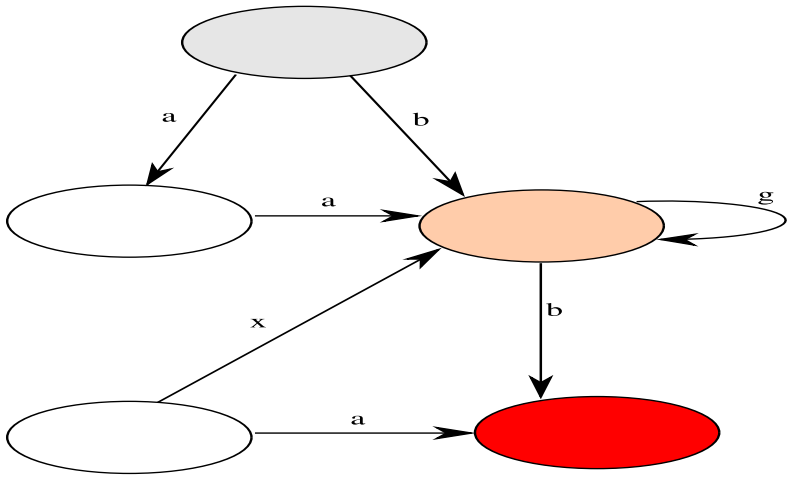
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



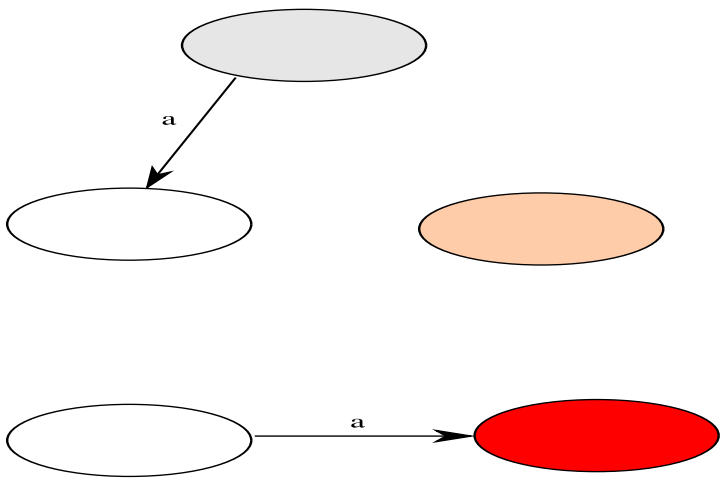
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



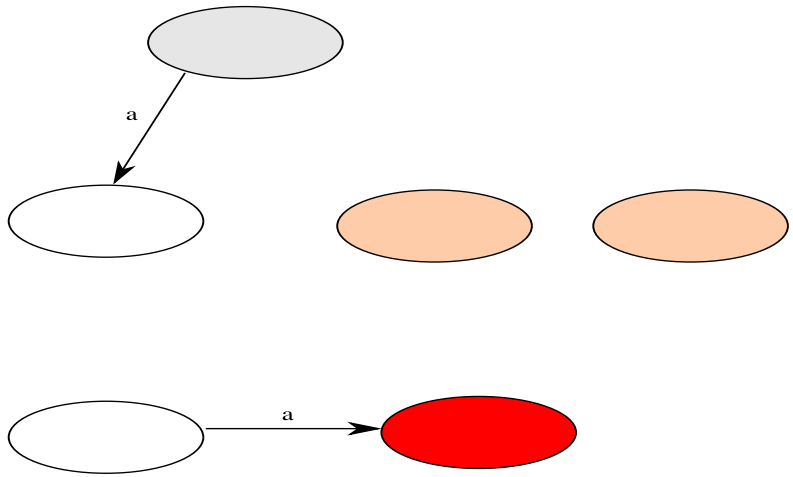
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



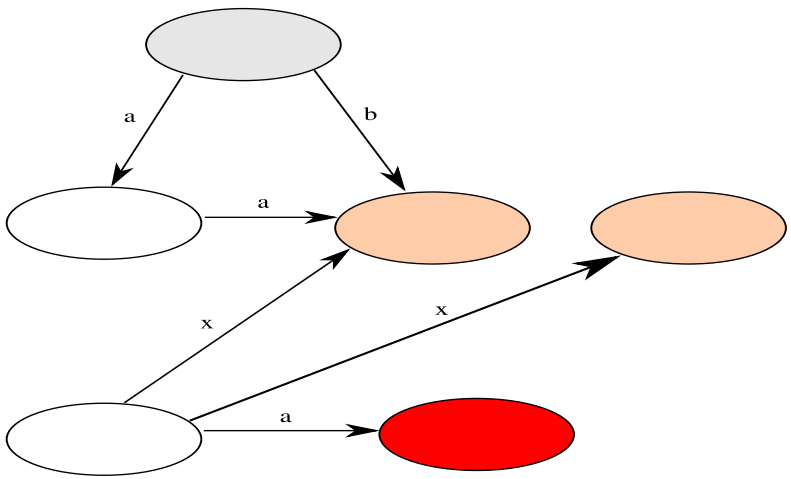
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



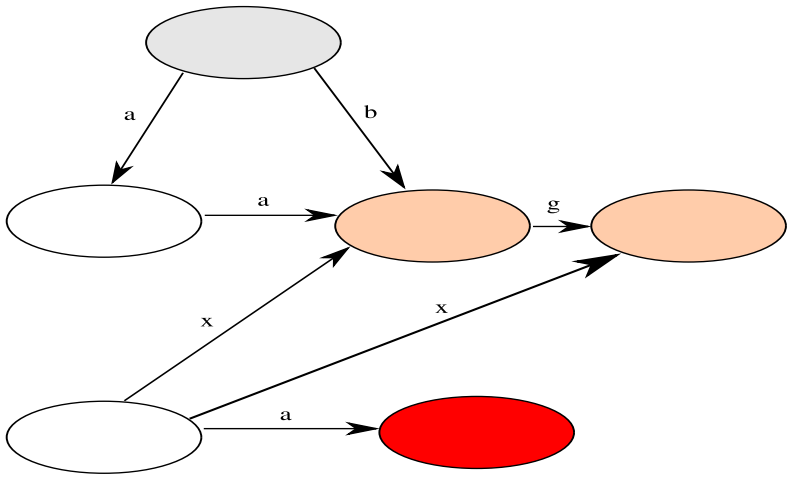
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



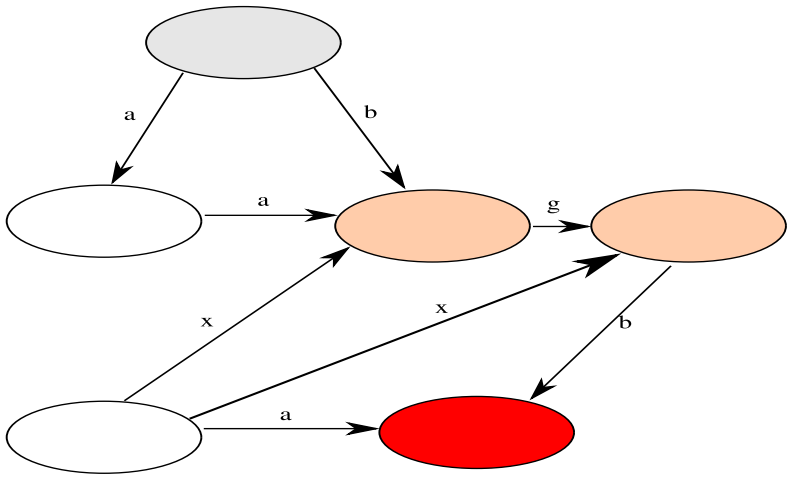
Recalcul de la relation de transition abstraite

Principe

On recalcule les transitions possibles autour des nœuds partagés.

Méthode et complexité

- Méthode : Pour toutes les transitions entrantes ou sortantes du nœud partagé, determiner la nouvelle source ou destination de la transition.
- Complexité : $\Theta(\text{degré}(q))$



Outline

Les méthodes de partage

Propriété

On n'atteint jamais une configuration où count = 4.

modèles	#Q	#R	# H_2	# H_1
foo 10	605	957	2	9
foo 100	51 005	81 507	2	99
foo 1000	5 010 005	8 018 007	2	999
foo 10000	500 100 005	800 150 007	2	9999
foo 100000	5001 000 005	8001 500 007	2	99999

Tab.: Benchmarks sur les programme foo

Les Model-Checker

Propriété

Si la porte d'un étage i est ouverte, alors la cage est à l'étage i.

modèle	#Q	#R	Durée	Mec	Arc
Lift 3	6 144	51 408	00:00:00	OK	OK
Lift 4	65 536	647 040	00:00:00	OK	OK
Lift 5	655 360	8 190 760	00:00:03	OK	OK
Lift 6	6 291 456	90 438 912	00:00:08	OK	KO
Lift 7	58 720 256	961 542 400	00:00:22	OK	KO
Lift 8	536 870 912	9 864 998 912	00:01:16	OK	KO
Lift 9	4 831 838 208	98 448 694 272	00:04:25	OK	KO
Lift 10	42 949 672 960	960 998 912 000	00:12:08	OK	KO
Lift 11	377 957 122 048	9 212 704 804 864	00:37:52	KO	KO

Tab.: Benchmarks sur les Lifts par la méthode CEGAR

Outline

- Extension aux modèles AltaRica sur domaines non-finis, et aux modèles temporisés.
- Vérification de propriétés de vivacité.
- Exploitation de la hiérarchie.
- Utilisation de modèle AltaRica comme abstraction.


André Arnold, Alain Griffault, Gérald Point, and Antoine Rauzy.
The altarica formalism for describing concurrent systems.
Fundamenta Informaticae, 40 :109–124, 2000.


T. Ball, R. Majumdar, T. Millstein, and S.K. Rajamani.
Automatic predicate abstraction of C programs.
ACM SIGPLAN Notices, 36(5) :203–213, 2001.


T. Ball, A. Podelski, and S.K. Rajamani.
Boolean and Cartesian abstraction for model checking C programs.
International Journal on Software Tools for Technology Transfer (STTT), 5(1) :49–58, 2003.


T. Ball, B. Cook, V. Levin, and S.K. Rajamani.

SLAM and Static Driver Verifier : Technology transfer of formal methods inside Microsoft.
Integrated Formal Methods, 2999 :1–20, 2004.


E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith.
Counterexample-guided abstraction refinement for symbolic model checking.
Journal of the ACM (JACM), 50(5) :752–794, 2003.


E.M. Clarke.
Model Checking.
MIT Press, 1999.


S. Graf and H. Saidi.
Construction of abstract state graphs of infinite systems with PVS.
Computer-Aided Verification (CAV), Lecture Notes in Computer Science. Springer, 1997.


A. Griffault and A. Vincent.
The Mec 5 model-checker.
International Conference on Computer Aided Verification (CAV), 2004.


T.A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre.
Software verification with Blast.
Proceedings of the 10th SPIN Workshop on Model Checking Software, pages 235–239, 2003.


W. Thomas.
Languages, automata, and logic, Handbook of formal languages, vol. 3 : beyond words.
1997.


Aymeric VINCENT.
Conception et réalisation d'un vérificateur de modèles AltaRica.

PhD thesis, Université Bordeaux 1, LaBRI, 2003.