SOUTENANCE DE THÈSE DE DOCTORAT

Discipline : *AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE*
Spécialité *: Automatique et Traitement du Signal*

Contribution à la commande des systèmes à événements discrets par filtre logique

Par **Romain PICHARD**

le 30 novembre 2018

Rapporteurs :

M. Jean-Marc ROUSSEL	Maître de conférences HDR, LURPA – ENS Paris-Saclay
M. Laurent PIÉTRAC	Maître de conférences HDR, AMPÈRE – INSA Lyon

Examinateurs :

Mme Véronique CARRÉ-MÉNÉTRIER	Professeur des Universités, CReSTIC – URCA
M. Jean-François PÉTIN	Professeur des Universités, CRAN – Université de Lorraine
M. Serge DEBERNARD	Professeur des Universités, LAMIH – Université Polytechnique Hauts-de-France

Co-encadrants de thèse :

Mme Ramla SADDEM	Maître de conférences, CReSTIC – URCA
M. Alexandre PHILIPPOT	Maître de conférences, CReSTIC – URCA

Invitée :

Mme Pascale MARANGÉ	Maître de conférences, CRAN – Université de Lorraine

Directeur de thèse :

M. Bernard RIERA	Professeur des Universités, CReSTIC – URCA

Table des matières

Contexte

Systèmes manufacturiers industriels

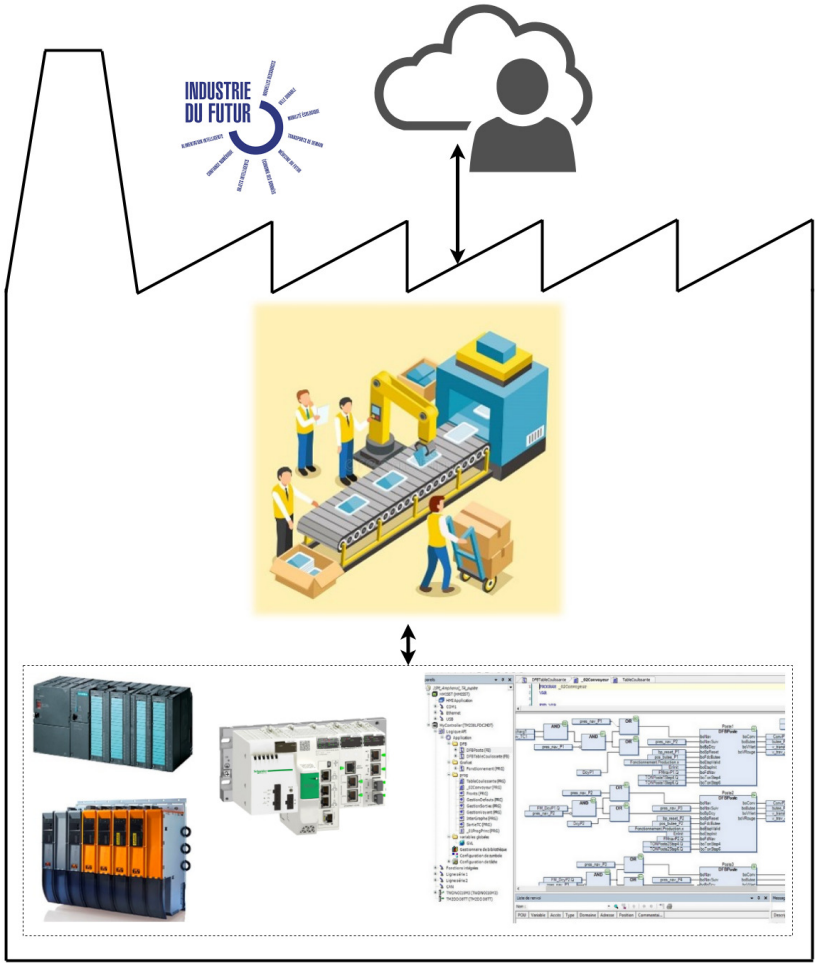
- fortement automatisés
- fonctionnement répétitif et séquentiel

Automate Programmable Industriel (API)

- fonctionnement normalisé (IEC 61131)
- faible capacité de calcul
- comportement cyclique et synchrone

Industrie du futur

- ouverture sur l'extérieur
- nécessité d'être flexible
- besoin de systèmes de production fiable



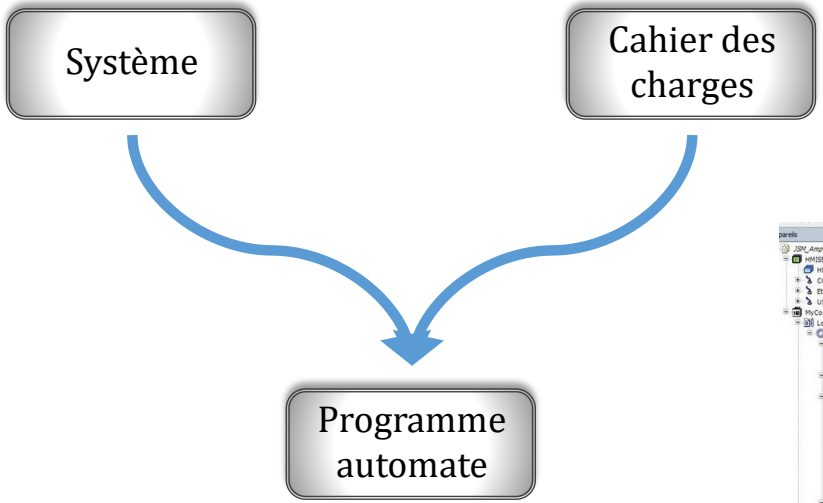
Problématique

Système

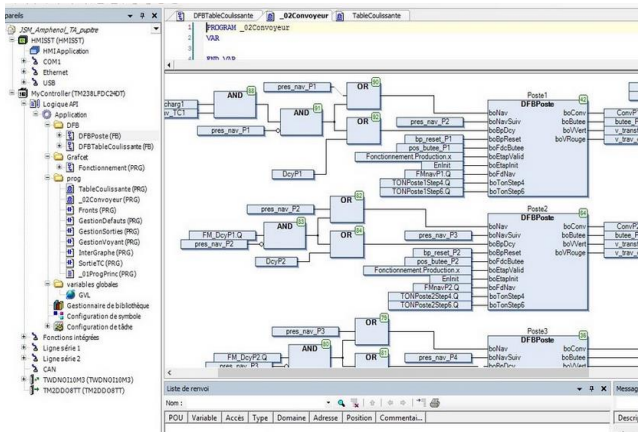
Cahier des charges

- Fonctionnel : Que veut-on faire avec le système ?
- Sécurité/sûreté : Que ne doit-on pas faire avec le système ?

Problématique

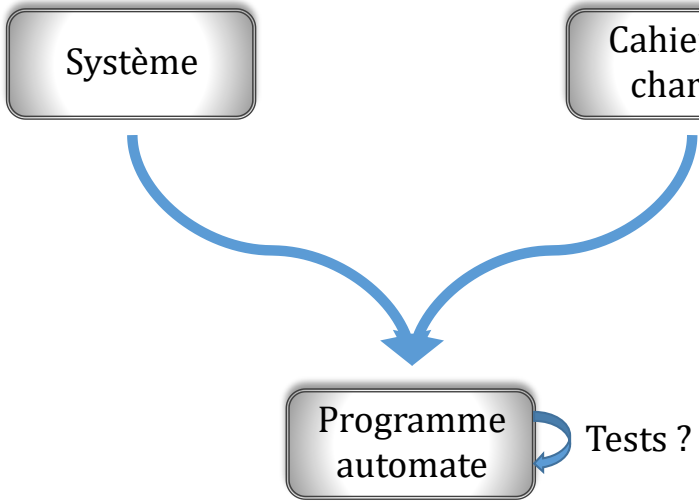


- Fonctionnel : Que veut-on faire avec le système ?
- Sécurité/sûreté : Que ne doit-on pas faire avec le système ?

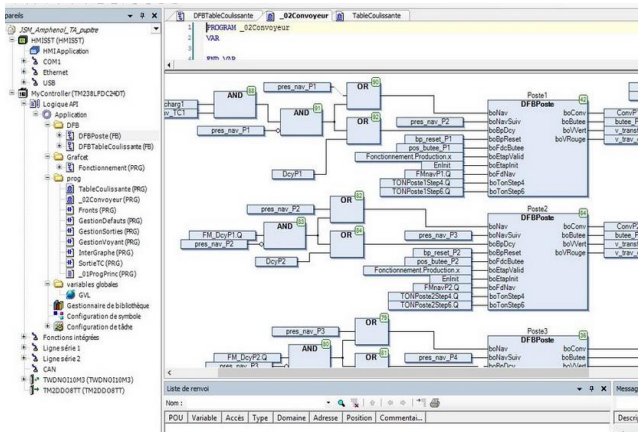


- Comment concevoir le programme automate ?

Problématique

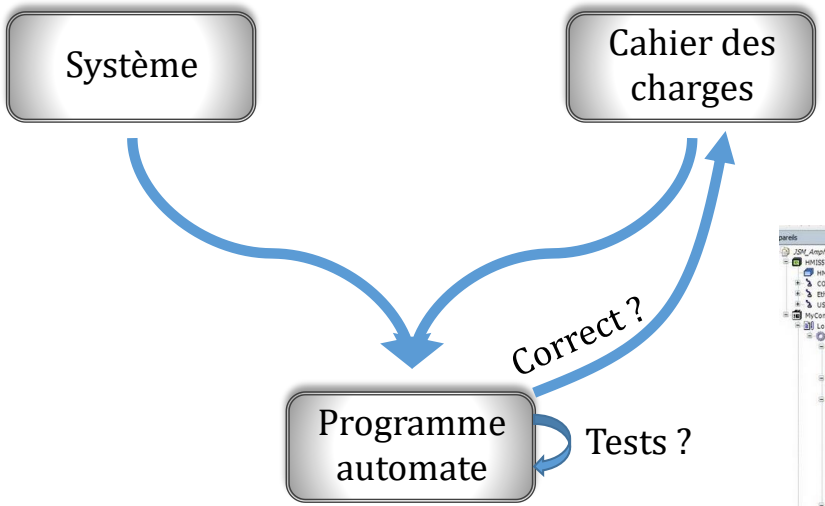


- Fonctionnel : Que veut-on faire avec le système ?
- Sécurité/sûreté : Que ne doit-on pas faire avec le système ?

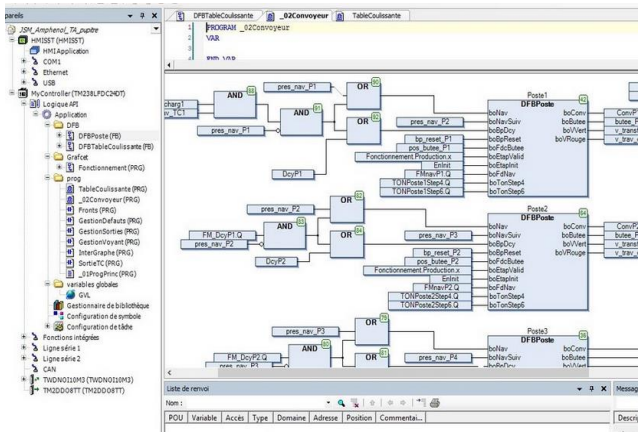


- Comment concevoir le programme automate ?
- Comment tester le programme ?

Problématique

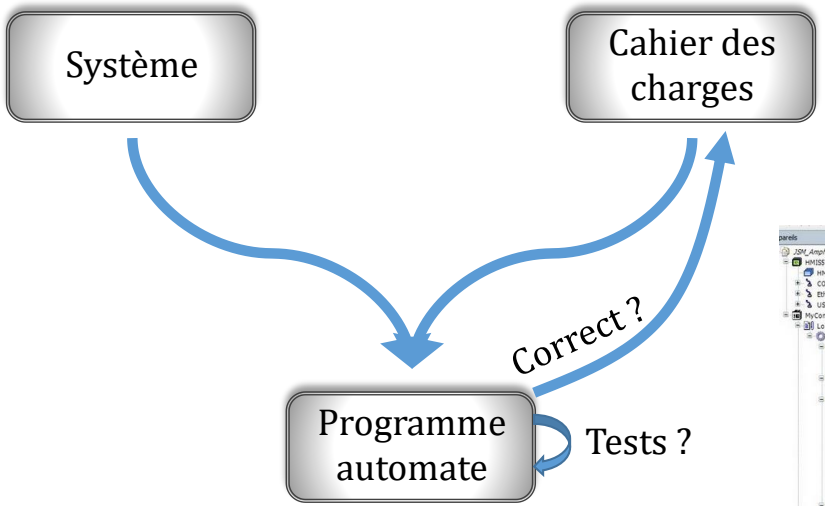


- Fonctionnel : Que veut-on faire avec le système ?
- Sécurité/sûreté : Que ne doit-on pas faire avec le système ?

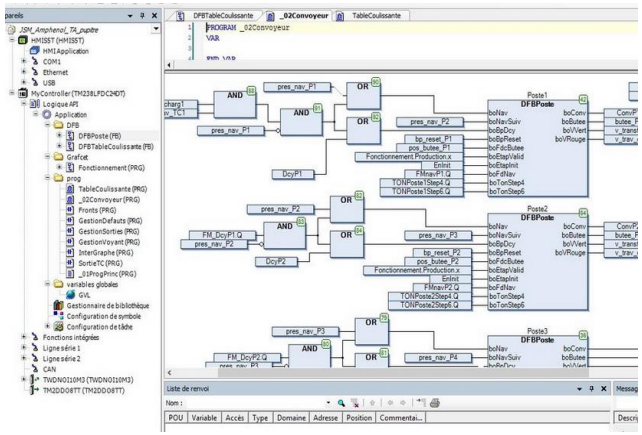


- Comment concevoir le programme automate ?
- Comment tester le programme ?
- Comment garantir que le programme répond à 100% au cahier des charges ?

Problématique



- Fonctionnel : Que veut-on faire avec le système ?
- Sécurité/sûreté : Que ne doit-on pas faire avec le système ?



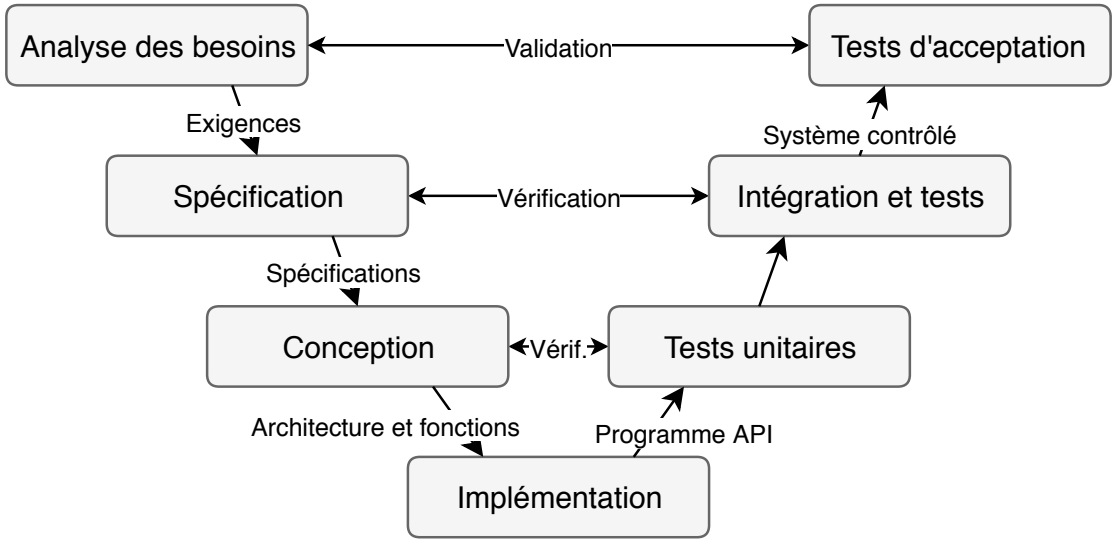
- Comment concevoir le programme automate ?
- Comment tester le programme ?
- Comment garantir que le programme répond à 100% au cahier des charges ?

METHODE(S) ?

Table des matières

Monde industriel

Approches pragmatiques

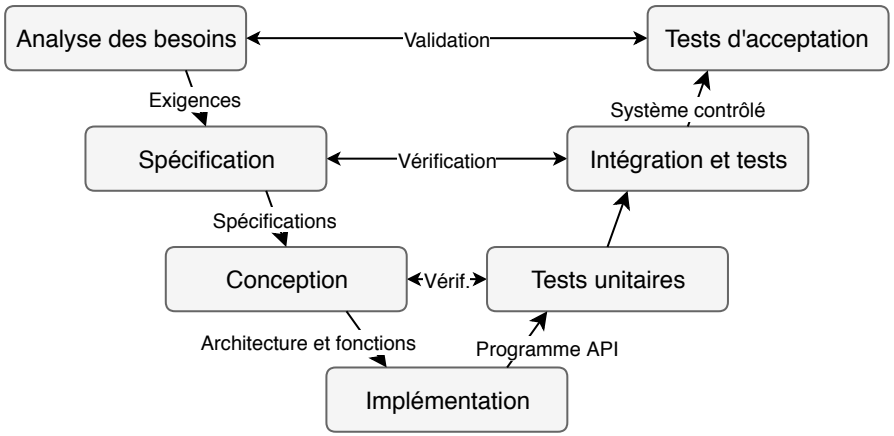


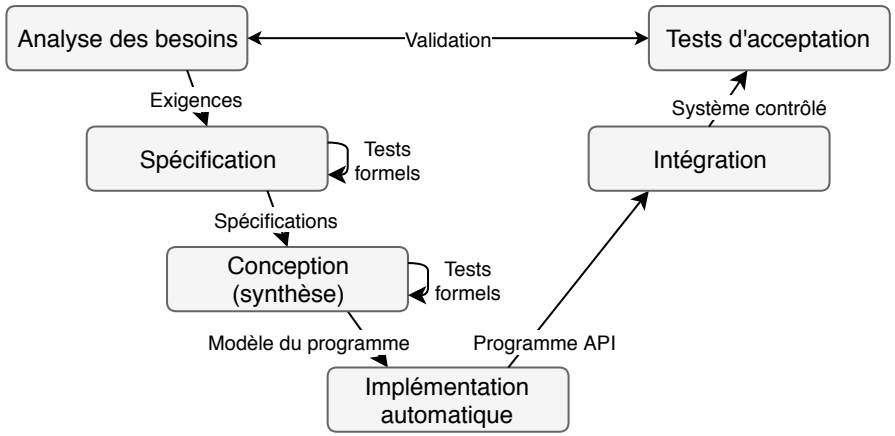
Approches pragmatiques

- Basées sur la pratique et l'expérience
- Amélioration successive du programme
- Conception/Implémentation/tests manuels
- Organisations cycliques (cycle en V, cycle itératif, approches agiles...)

Monde académique

Approches formelles





Approches formelles

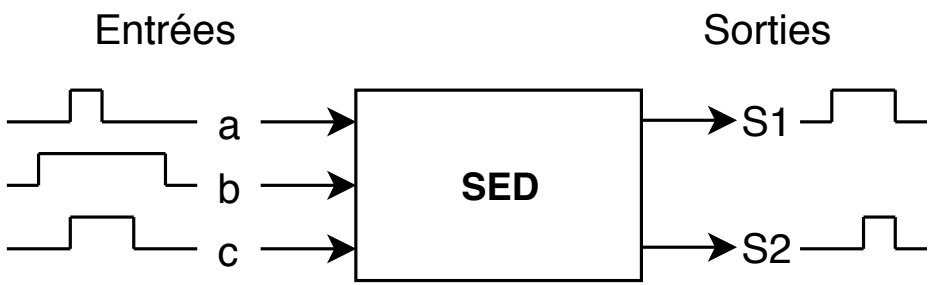
Objectifs : automatiser la conception, les tests et l'implémentation

- Formalismes mathématiques pour les spécifications
- Théorèmes et propriétés pour les tests
- Génération automatique du résultat

Systèmes à événements discrets

Systèmes à événements discrets (SED) (Cassandras et Lafortune, 2009)

- Systèmes dont l'état interne évolue en fonction de l'occurrence d'événements
- Un événement se produit instantanément et cause la transition de l'état d'une valeur (discrète) à une autre valeur
- Formalismes usuels : théorie des langages, automates à états finis, réseaux de Petri, algèbre de Boole, algèbre (max,+)



Systèmes à événements discrets

Systèmes à événements discrets (SED) (Cassandras et Lafortune, 2009)

- Systèmes dont l'état interne évolue en fonction de l'occurrence d'événements
- Un événement se produit instantanément et cause la transition de l'état d'une valeur (discrète) à une autre valeur
- Formalismes usuels : théorie des langages, automates à états finis, réseaux de Petri, algèbre de Boole, algèbre (max,+)

Commande d'un SED (Brandin et Wonham, 1994)

- Réalisé par un contrôleur
- Impose une évolution au système : force l'occurrence d'événements
- Événements commandables

Systèmes à événements discrets

Systèmes à événements discrets (SED) (Cassandras et Lafortune, 2009)

- Systèmes dont l'état interne évolue en fonction de l'occurrence d'événements
- Un événement se produit instantanément et cause la transition de l'état d'une valeur (discrète) à une autre valeur
- Formalismes usuels : théorie des langages, automates à états finis, réseaux de Petri, algèbre de Boole, algèbre (max,+)

Commande d'un SED (Brandin et Wonham, 1994)

- Réalisé par un contrôleur
- Impose une évolution au système : force l'occurrence d'événements
- Événements commandables

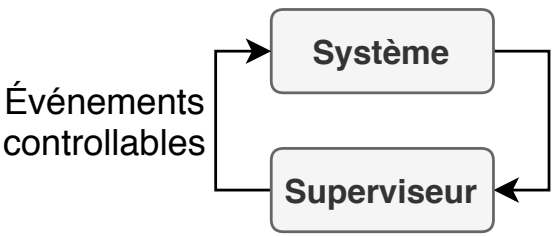
Contrôle d'un SED (Ramadge et Wonham, 1989)

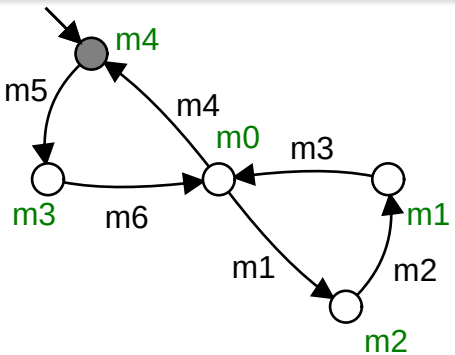
- Réalisé par un superviseur
- Restreint l'évolution du système : interdit l'occurrence d'événements
- Événements contrôlables

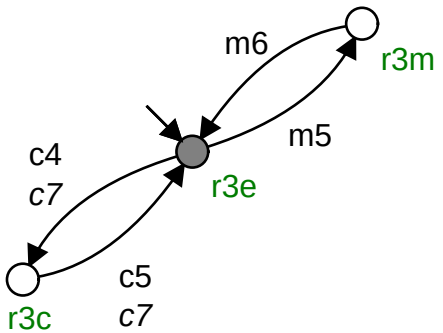
Théorie de la commande par supervision (SCT)

Caractéristiques (Ramadge et Wonham, 1989)

- Spécification à l'aide d'automates à états finis
- Synthèse formel d'un superviseur sous forme d'un automate à états finis



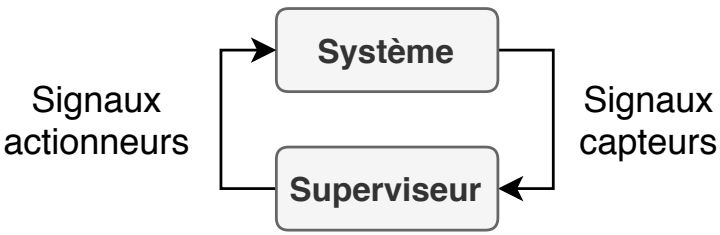


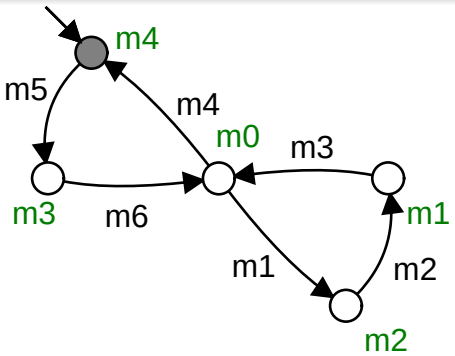


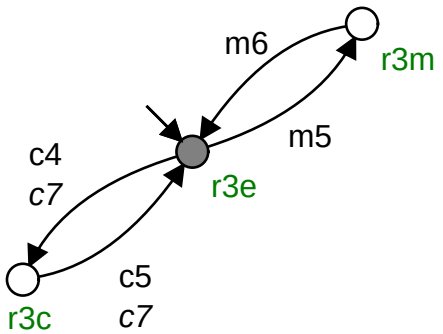
Théorie de la commande par supervision (SCT)

SCT pour la commande par API

- Signaux capteurs : événements observables et non-contrôlables
- Signaux actionneurs : événements contrôlables







Avantages

- Théorie majoritairement étudiée
- Représentation graphique du résultat

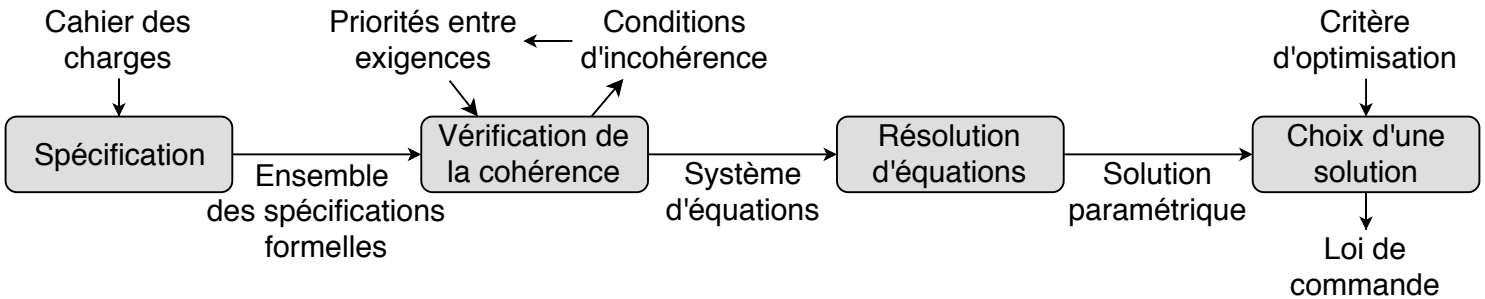
Inconvénients (Zaytoon et Riera, 2017)

- Risque d'explosion combinatoire
- Traduction du superviseur en contrôleur non trivial
- Difficilement utilisable avec un API

Synthèse algébrique

Caractéristiques (Hietter, 2009)

- Spécification à l'aide d'équations logiques
- Synthèse d'un contrôleur sous forme d'équations logiques
- Solution calculée en amont de l'implémentation



Avantages

- Couvre l'ensemble du processus de développement
- Implémentation et intégration très facile

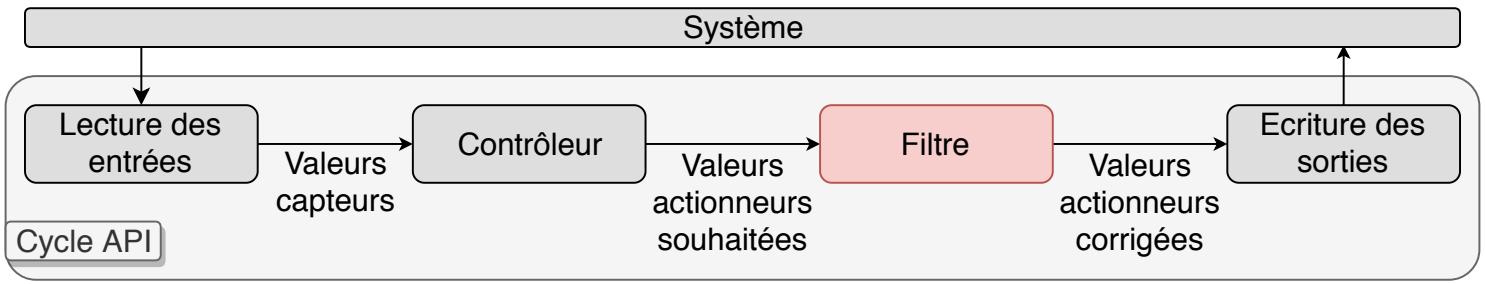
Inconvénients

- Risque d'explosion combinatoire
- Comportement du contrôleur dépend du critère
- Équations finales difficiles à lire

Filtre logique de sécurité

Caractéristiques (Marangé, 2008)

- Spécification à l'aide d'équations logiques appelées **contraintes de sécurité**
- Synthèse d'un filtre logique de commande
- Permet de trouver une solution dynamiquement pendant l'exécution du système



Avantages

- Faible risque d'explosion combinatoire
- Implémentation et intégration facile

Inconvénients

- Nécessité d'imposer des priorités entre les actionneurs
- Comportement dépendant de l'algorithme d'implémentation

Constats

Contexte

- Usine du futur : besoin de flexibilité et de fiabilité
- API : fonctionnement normé et faible capacité de calcul

Problématique

Méthodes de conception de programme API ?

Constats

Contexte

- Usine du futur : besoin de flexibilité et de fiabilité
- API : fonctionnement normé et faible capacité de calcul

Problématique

Méthodes de conception de programme API ?

Constats

Monde industriel : méthodes basées sur l'expérience et les tests

- obtention d'un résultat rapidement
- vérification/validation non-exhaustive

Monde académique : méthodes formelles

- temps de spécification long mais vérification exhaustive et implémentation automatique
- peu utilisées dans l'industrie : fort risque d'explosion combinatoire

Constats

Contexte

- Usine du futur : besoin de flexibilité et de fiabilité
- API : fonctionnement normé et faible capacité de calcul

Problématique

Méthodes de conception de programme API ?

Constats

Monde industriel : méthodes basées sur l'expérience et les tests

- obtention d'un résultat rapidement
- vérification/validation non-exhaustive

Monde académique : méthodes formelles

- temps de spécification long mais vérification exhaustive et implémentation automatique
- peu utilisées dans l'industrie : fort risque d'explosion combinatoire

SCT

\+ Formalisme graphique
\- Explosion combinatoire
\- Asynchronisme événements

Synthèse algébrique

\+ Équations logiques
\- Explosion combinatoire
\- Lisibilité solution

Filtre de sécurité

\+ Équations logiques
\- Formalisme/généralisation
\- Méthodologie

Travaux précédents

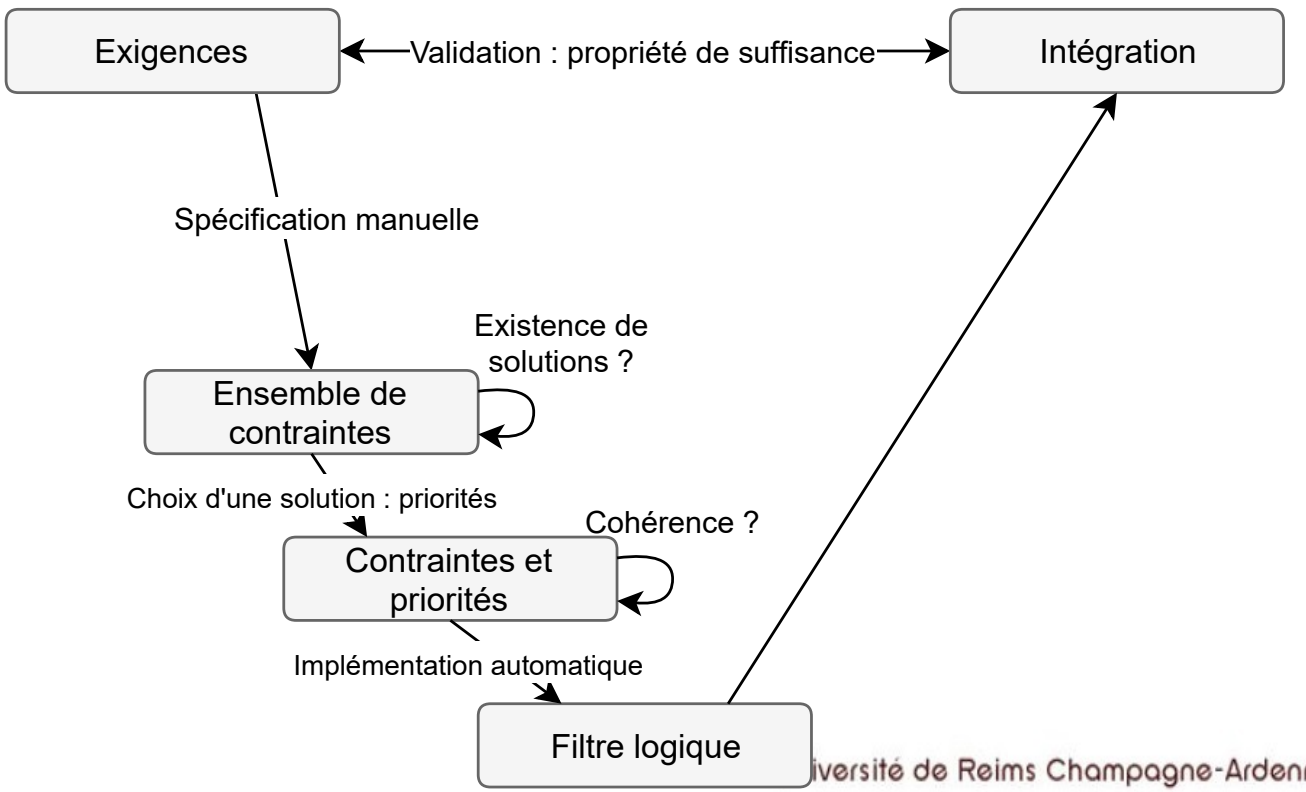
- Marangé, 2008
  - Proposition de l'approche par filtre de sécurité
  - Vérification de la propriété de **suffisance** du filtre de sécurité par model-checking
  - Algorithme de résolution "bloquant"
- Benlorhfar, 2011
  - Algorithme de résolution "correcteur"
  - La résolution dépend de l'ordre d'écriture des contraintes
- Coupat, 2014
  - Algorithme "itératif"
  - Proposition de la notion de **priorité**
  - Contraintes combinées limitées à 2 variables commandables
- Riera, 2015
  - Proposition de la notion de **cohérence** du filtre
  - Proposition de 3 modes d'utilisations du filtre (bloquant, superviseur, contrôleur)
  - Conditions uniquement **nécessaires** à la cohérence

Objectifs

- Généraliser l'approche par filtre logique : formalisme, méthodologie
- Proposer une condition nécessaire et suffisante pour vérifier la cohérence
- Proposer des algorithmes d'implémentation plus flexibles

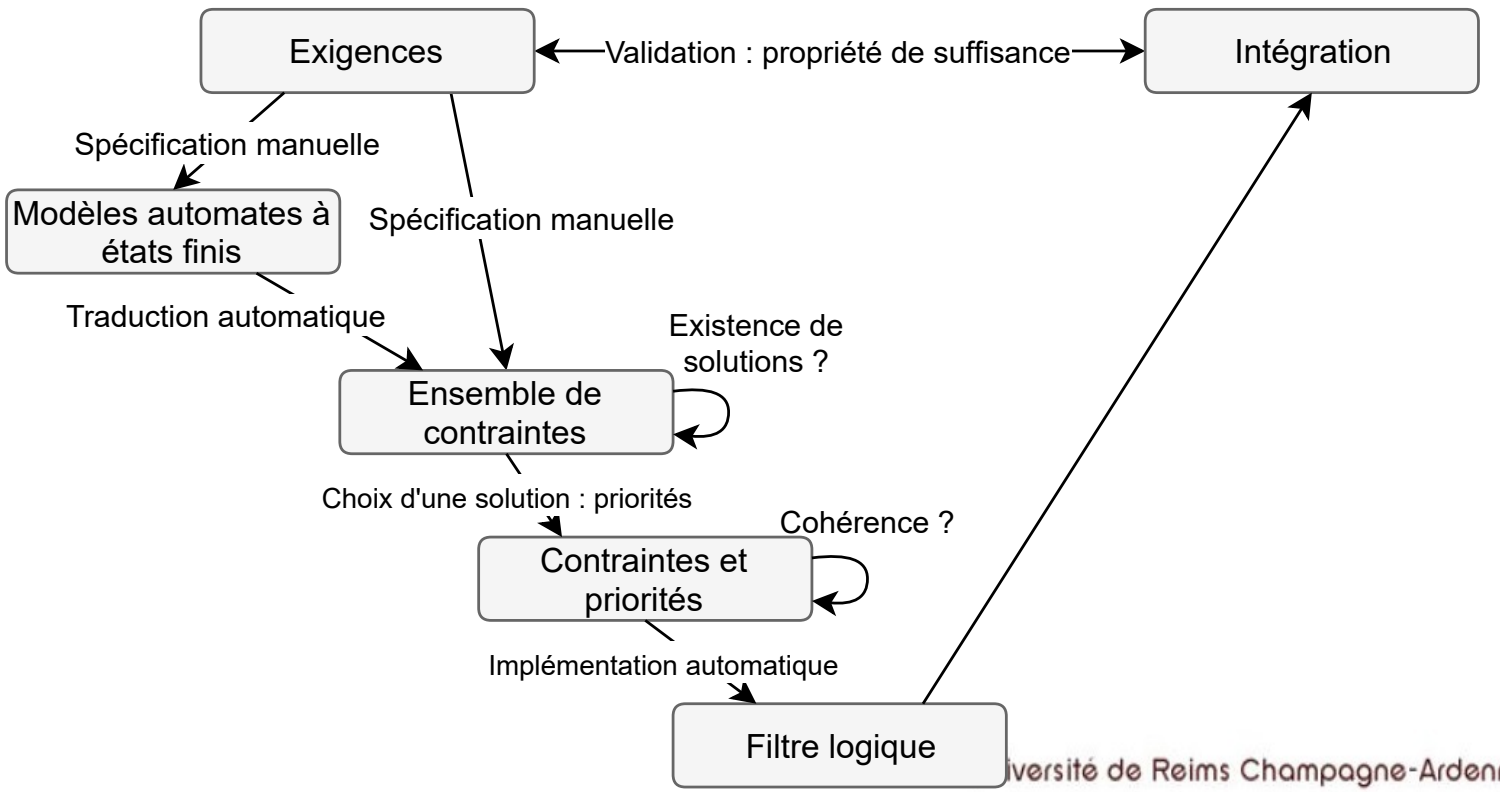
Contributions

- Proposition d'une démarche globale de conception du filtre logique
- Formalisation des définitions et propriétés liées au filtre logique
- Formalisation de la cohérence d'un ensemble de contraintes et proposition d'une condition nécessaire et suffisante à sa vérification
- Proposition de différents algorithmes d'implémentation du filtre dans un API



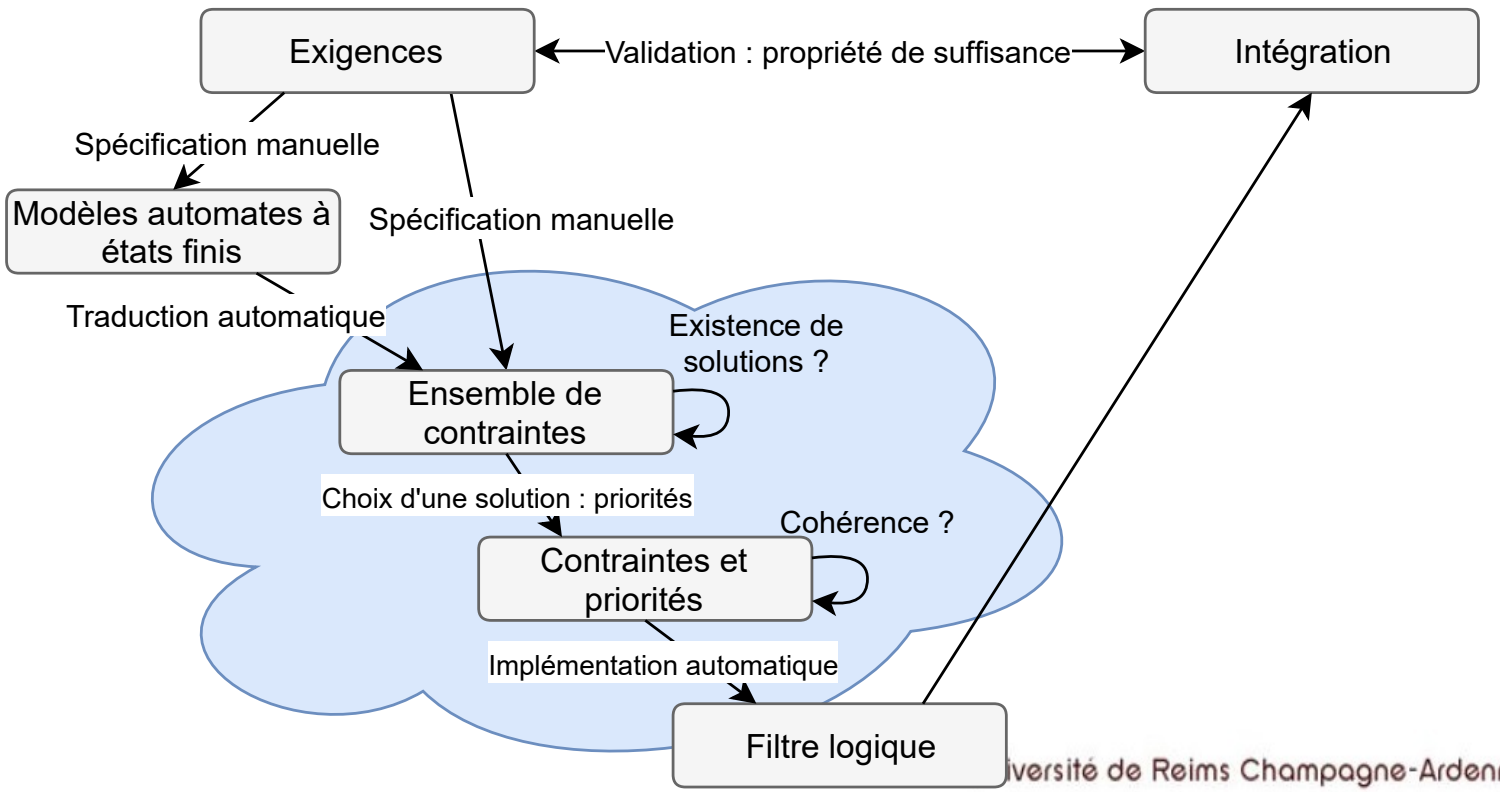
Contributions

- Proposition d'une démarche globale de conception du filtre logique
- Formalisation des définitions et propriétés liées au filtre logique
- Formalisation de la cohérence d'un ensemble de contraintes et proposition d'une condition nécessaire et suffisante à sa vérification
- Proposition de différents algorithmes d'implémentation du filtre dans un API
- Proposition d'une approche de génération automatique des contraintes logiques



Contributions

- Proposition d'une démarche globale de conception du filtre logique
- Formalisation des définitions et propriétés liées au filtre logique
- Formalisation de la cohérence d'un ensemble de contraintes et proposition d'une condition nécessaire et suffisante à sa vérification
- Proposition de différents algorithmes d'implémentation du filtre dans un API
- Proposition d'une approche de génération automatique des contraintes logiques
- Développement d'un outil logiciel : automatiser les analyses et l'implémentation



Contributions

- Proposition d'une démarche globale de conception du filtre logique
- Formalisation des définitions et propriétés liées au filtre logique
- Formalisation de la cohérence d'un ensemble de contraintes et proposition d'une condition nécessaire et suffisante à sa vérification
- Proposition de différents algorithmes d'implémentation du filtre dans un API
- Proposition d'une approche de génération automatique des contraintes logiques
- Développement d'un outil logiciel : automatiser les analyses et l'implémentation

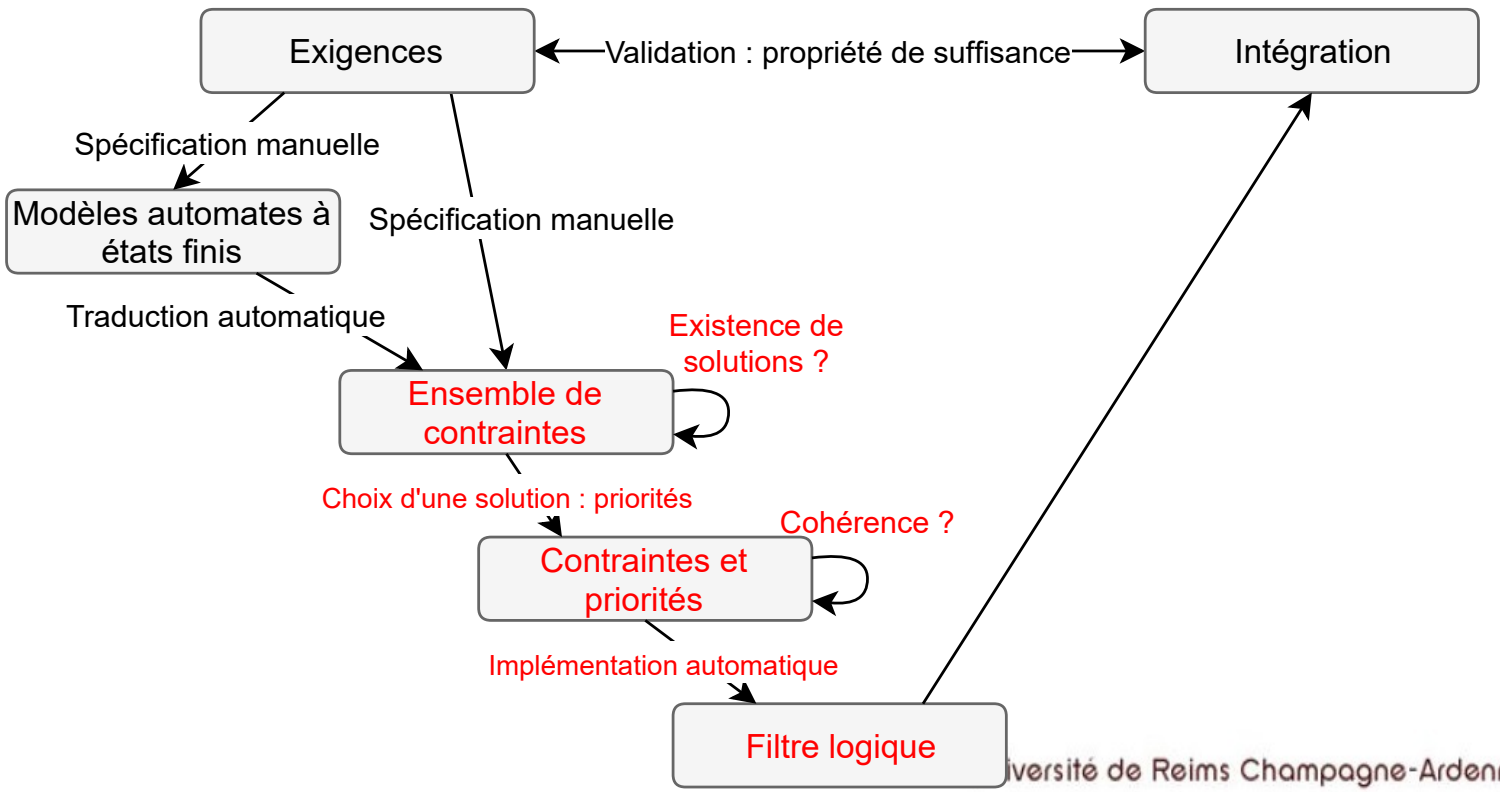
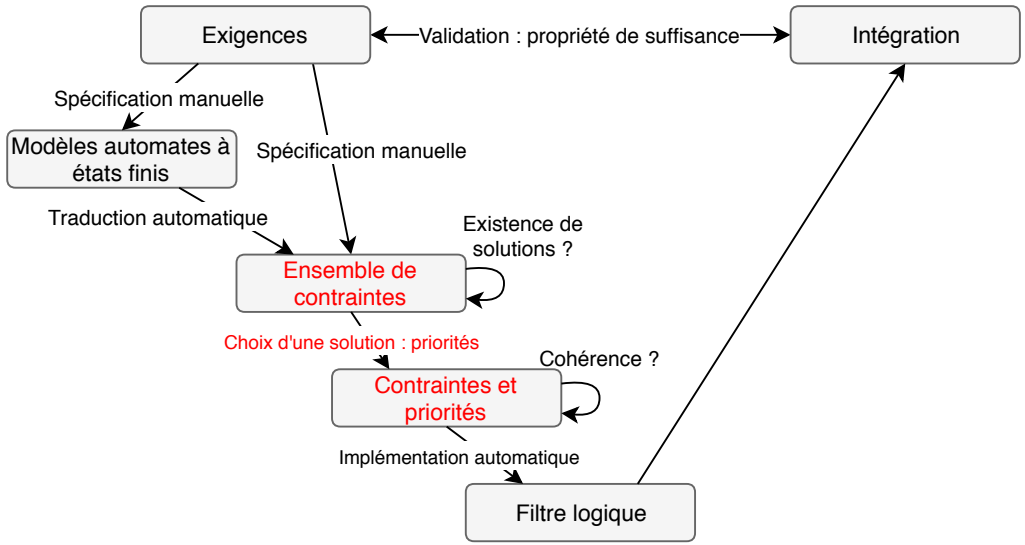
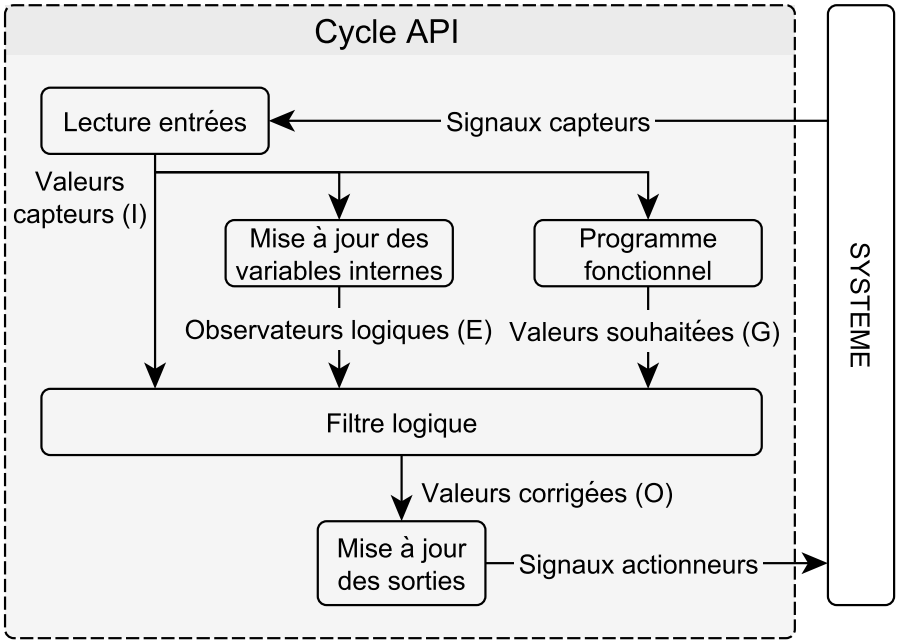


Table des matières



Définitions

Filtre logique



Filtre logique : composition

- Variables commandables (O) et non-commandables ($Y = G \cup I \cup E$) ;
- Contraintes logiques ;
- Priorités fonctionnelles.

Filtre logique : objectif

Tester et corriger les valeurs souhaitées (G) afin de respecter l'ensemble des contraintes logiques.

Contraintes logiques (Déf. 17 p102)

- Une contrainte logique est définie comme un **produit** de littéraux ;
- Un littéral est une variable logique ou son complémentaire ;

$$C_n : \begin{array}{ccc} \overset{*}{2^Y} \times \overset{*}{2^O} & \longrightarrow & \Gamma \\ (\overset{*}{Y^{C_n}}, \overset{*}{O^{C_n}}) & \longmapsto & \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) = Y^{C_n} \cdot O^{C_n} \end{array} , \forall n \in \{1, \ldots, N_c\}$$

Contrainte logique satisfaite

- Une contrainte logique est dite **satisfaite** si celle-ci est évaluée à **faux**.

$$C_n = \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) = 0$$

Contrainte logique violée

- Une contrainte logique est dite **violée** si celle-ci est évaluée à **vraie**.

$$C_n = \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) = 1$$

Définitions

Contraintes

$$C_n : \begin{array}{ccc} \overset{*}{2^Y} \times \overset{*}{2^O} & \longrightarrow & \Gamma \\ (\overset{*}{Y^{C_n}}, \overset{*}{O^{C_n}}) & \longmapsto & \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) \end{array} , \forall n \in \{1, \ldots, N_c\}$$

Types de contraintes (Déf. 19 à 21 p103)

$Dim(\overset{*}{O^{C_n}}) = 0$: contraintes structurelles (C_{struct})

- produit de variables non-commandables ;
- utilisées pour exprimer des états non-atteignables par le système.

Définitions

Contraintes

$$C_n : \begin{array}{ccc} \overset{*}{2^Y} \times \overset{*}{2^O} & \longrightarrow & \Gamma \\ (\overset{*}{Y^{C_n}}, \overset{*}{O^{C_n}}) & \longmapsto & \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) \end{array} , \forall n \in \{1, \ldots, N_c\}$$

Types de contraintes (Déf. 19 à 21 p103)

$Dim(\overset{*}{O^{C_n}}) = 0$: contraintes structurelles (C_{struct})

- produit de variables non-commandables ;
- utilisées pour exprimer des états non-atteignables par le système.

$Dim(\overset{*}{O^{C_n}}) = 1$: contraintes simples (C_s) :

- produit de variables non-commandables et une seule variable commandable ;
- représente une exigence de type "Si ... alors ...".

Définitions

Contraintes

$$C_n : \begin{array}{ccc} \overset{*}{2^{Y}} \times \overset{*}{2^{O}} & \longrightarrow & \Gamma \\ (\overset{*}{Y^{C_n}}, \overset{*}{O^{C_n}}) & \longmapsto & \prod(\overset{*}{Y^{C_n}}) \cdot \prod(\overset{*}{O^{C_n}}) \end{array} , \forall n \in \{1, \ldots, N_c\}$$

Types de contraintes (Déf. 19 à 21 p103)

$Dim(\overset{*}{O^{C_n}}) = 0$: contraintes structurelles (C_{struct})

- produit de variables non-commandables ;
- utilisées pour exprimer des états non-atteignables par le système.

$Dim(\overset{*}{O^{C_n}}) = 1$: contraintes simples (C_s) :

- produit de variables non-commandables et une seule variable commandable ;
- représente une exigence de type "Si ... alors ...".

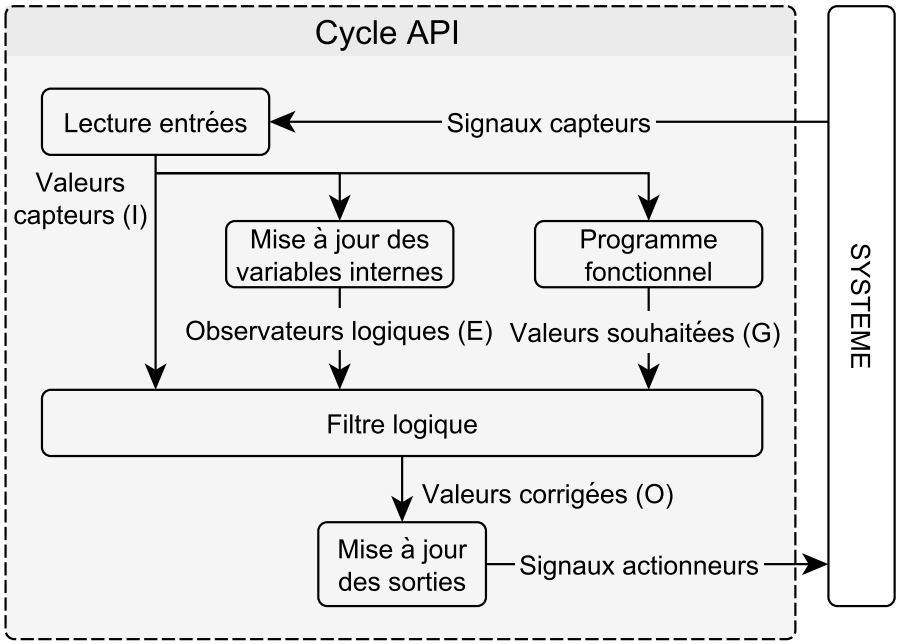
$Dim(\overset{*}{O^{C_n}}) > 1$: contraintes combinées (C_c) :

- produit de variables non-commandables et plusieurs variables commandables ;
- représente une exigence d'exclusion mutuelle entre variables.

Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cs_3 = a \cdot \overline{O_3}$

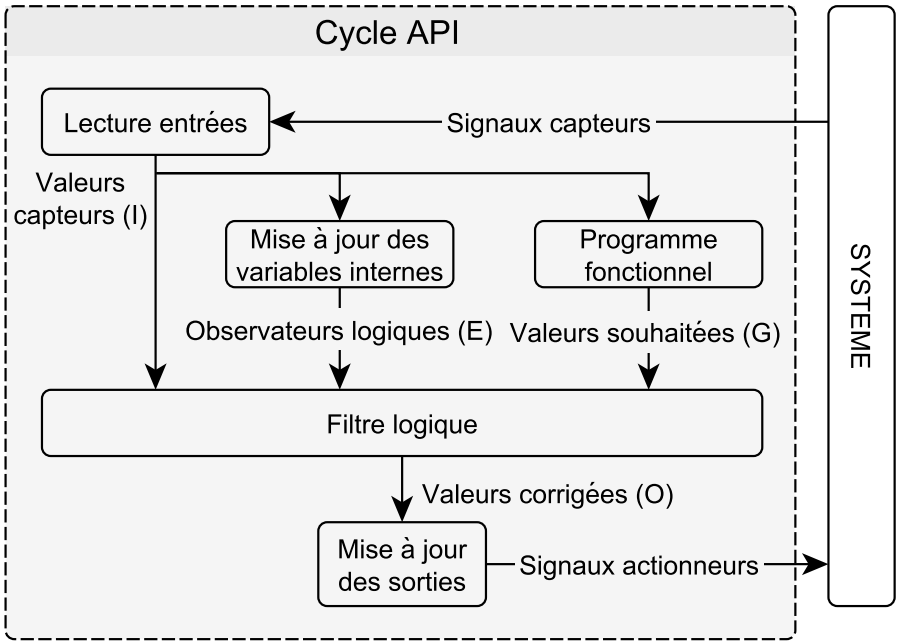
- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3}$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cs_3 = a \cdot \overline{O_3}$

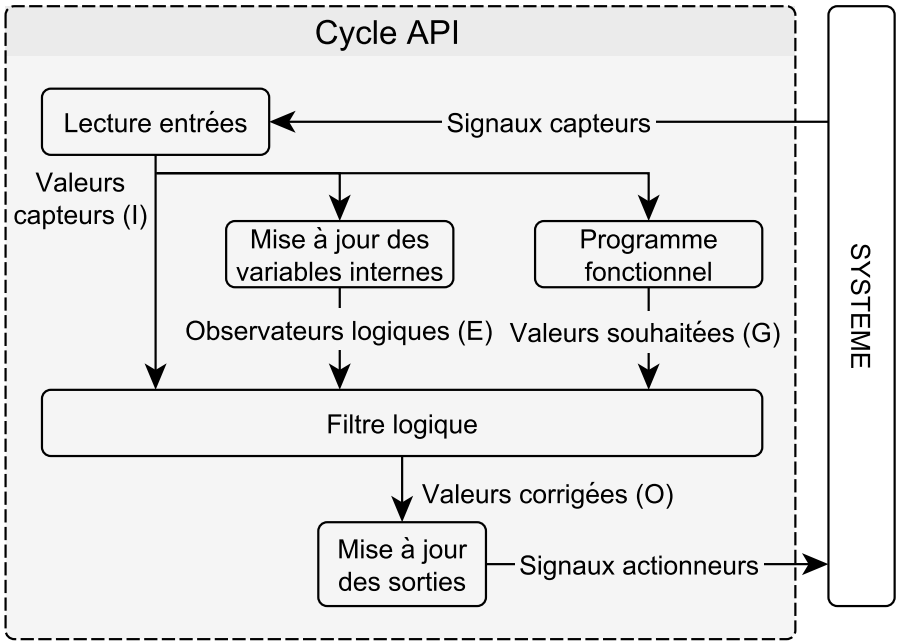
- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3}$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot g_2$
- $Cs_2 = \overline{a} \cdot b \cdot g_1$
- $Cs_3 = a \cdot \overline{g_3}$

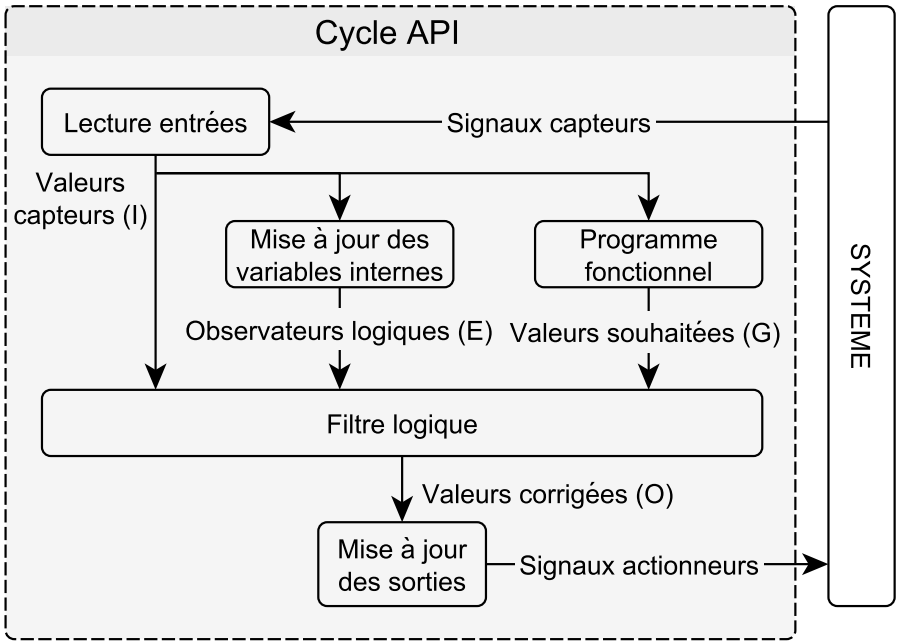
- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3}$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1 \quad \Longrightarrow O_1 = 0$
- $Cs_3 = a \cdot \overline{O_3}$

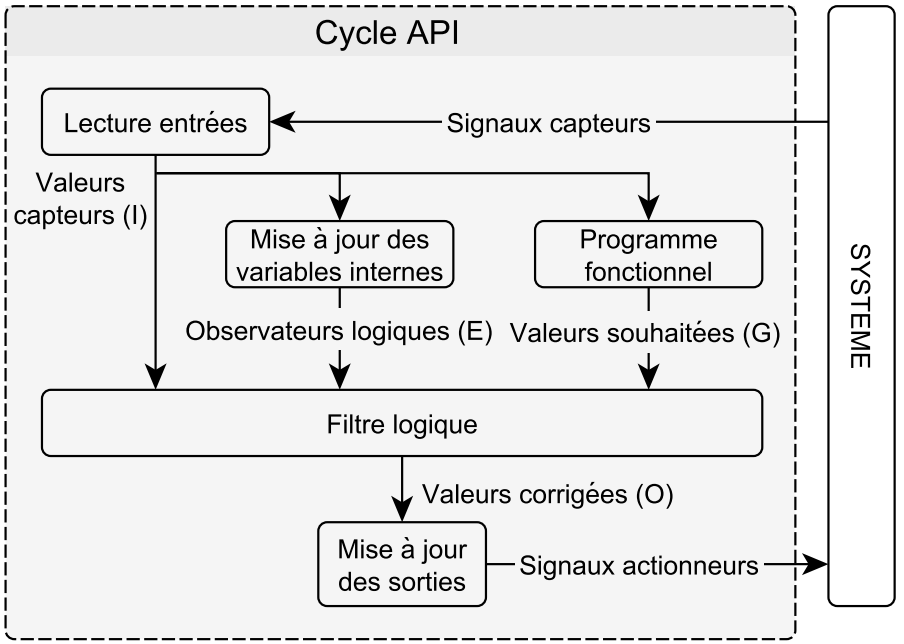
- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3}$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1 \quad \Longrightarrow O_1 = 0$
- $Cs_3 = a \cdot \overline{O_3}$

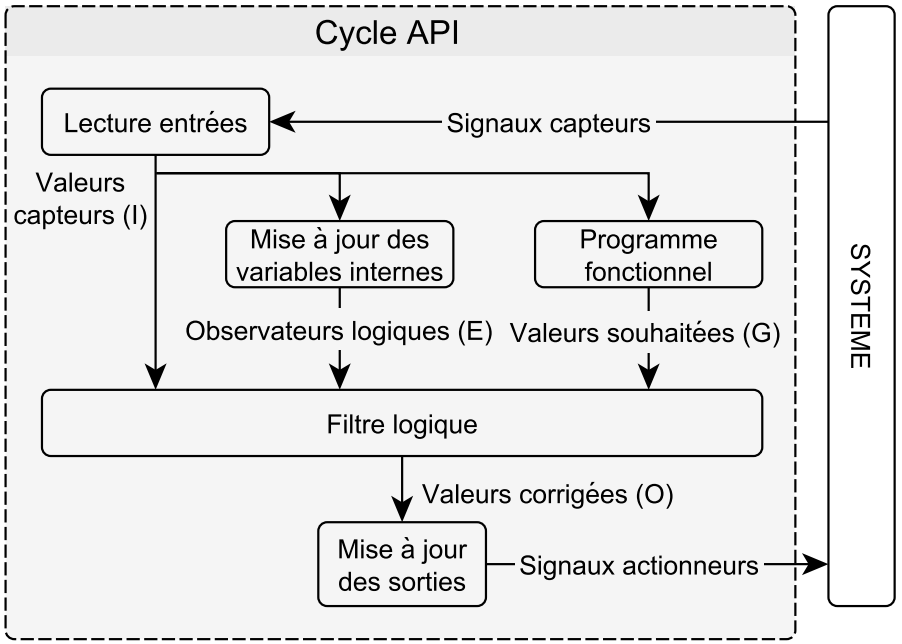
- $Cc_1 = O_1 \cdot \overline{g_2}$
- $Cc_2 = g_2 \cdot \overline{g_3}$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1 \Longrightarrow O_1 = 0$
- $Cs_3 = a \cdot \overline{O_3}$

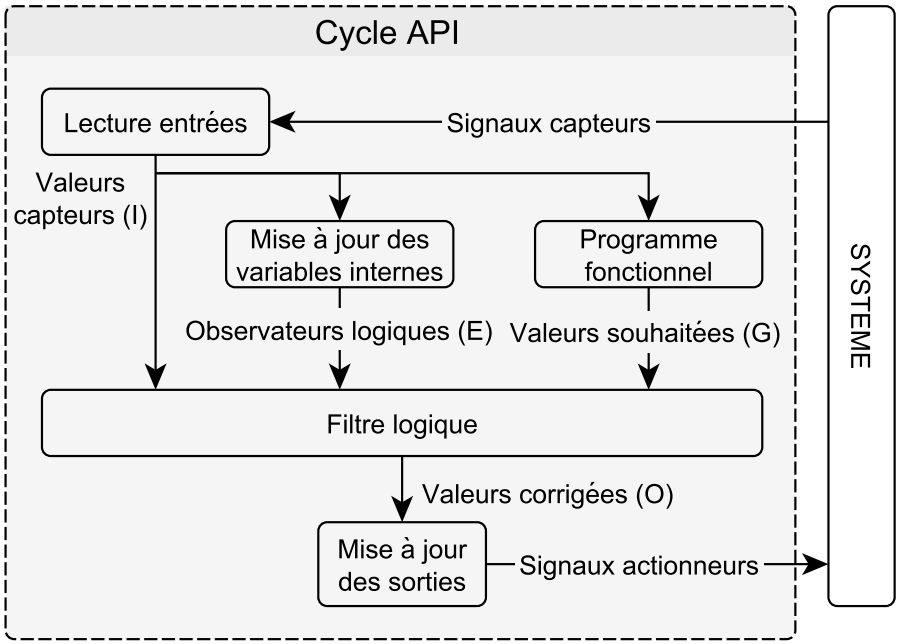
- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3} \Longrightarrow O_3 = 1$



Résolution des contraintes

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1 \Longrightarrow O_1 = 0$
- $Cs_3 = a \cdot \overline{O_3}$

- $Cc_1 = O_1 \cdot \overline{O_2}$
- $Cc_2 = O_2 \cdot \overline{O_3} \Longrightarrow O_3 = 1$



Calcul du vecteur de sortie

	O_1	O_2	O_3
G	1	1	0
M	0	X	1
	-	-	-
O	0	1	1

Résolution des contraintes simples

$$Cs_i^k = \prod(Y_i^k) \cdot \overset{*}{o}_k = \begin{cases} \prod(Y0_i^k) \cdot o_k \\ OU \\ \prod(Y1_i^k) \cdot \overline{o_k} \end{cases}$$

Résolution des contraintes simples

$$Cs_i^k = \prod(Y_i^k) \cdot \overset{*}{o_k} = \begin{cases} \prod(Y0_i^k) \cdot o_k & \Longrightarrow o_k = 0 \\ OU & \\ \prod(Y1_i^k) \cdot \overline{o_k} & \Longrightarrow o_k = 1 \end{cases}$$

Résolution des contraintes simples

$$Cs_i^k = \prod(Y_i^k) \cdot \overset{*}{o_k} = \begin{cases} \prod(Y0_i^k) \cdot o_k & \Longrightarrow o_k = 0 \\ OU & \\ \prod(Y1_i^k) \cdot \overline{o_k} & \Longrightarrow o_k = 1 \end{cases}$$

Fonctions de forçage simples (p108)

- Polynômes logiques composés uniquement de variables non-commandables ;
- Permet de savoir si une variable commandable (O_k) doit être forcée à 0 ($F0s^k$) ou à 1 ($F1s^k$) pour résoudre une ou plusieurs contraintes simples ;
- Construits à partir des parties non-commandables des contraintes simples ;

$$F0s^k = \sum_{i=1}^{Ncs_1^k}(\prod(Y0_i^k)) \qquad F1s^k = \sum_{i=1}^{Ncs_1^k}(\prod(Y1_i^k))$$

Résolution des contraintes simples

$$Cs_i^k = \prod(Y_i^k) \cdot \overset{*}{o_k} = \begin{cases} \prod(Y0_i^k) \cdot o_k & \Longrightarrow o_k = 0 \\ OU & \\ \prod(Y1_i^k) \cdot \overline{o_k} & \Longrightarrow o_k = 1 \end{cases}$$

Fonctions de forçage simples (p108)

- Polynômes logiques composés uniquement de variables non-commandables ;
- Permet de savoir si une variable commandable (O_k) doit être forcée à 0 ($F0s^k$) ou à 1 ($F1s^k$) pour résoudre une ou plusieurs contraintes simples ;
- Construits à partir des parties non-commandables des contraintes simples ;

$$F0s^k = \sum_{i=1}^{Ncs_1^k}(\prod(Y0_i^k)) \qquad F1s^k = \sum_{i=1}^{Ncs_1^k}(\prod(Y1_i^k))$$

Exemple

$$\left.\begin{array}{l} Cs_1^1 = a \cdot O_1 \\ Cs_2^1 = \overline{b} \cdot O_1 \end{array}\right\} F0s^1 = a + \overline{b}$$

$$\left.\begin{array}{l} Cs_3^1 = b \cdot \overline{O_1} \end{array}\right\} F1s^1 = b$$

Résolution des contraintes combinées

Problématique

- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \implies \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 \\ O_1 := 0 & \text{et} \quad O_2 := O_2 \\ O_1 := O_1 & \text{et} \quad O_2 := 1 \end{cases}$
- Contrainte combinée à n variables commandables : $2^n - 1$ possibilités ;
- Résolution déterministe : un choix doit être fait ;

Solution

- Notion de priorité entre variables commandables ;
- Deux niveaux de priorités : structurelles et fonctionnelles ;

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Exemple

- $Cs_1 = a \cdot O_2 \qquad \Longrightarrow O_2 := 0 \ (F0s^2 = a)$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1$

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow O_2 := 0 \ (F0s^2 = a)$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \quad \Longrightarrow O_1 := 0$

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow O_2 := 0 \ (F0s^2 = a)$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \quad \Longrightarrow O_1 := 0$
- $prioS_{Cc_1} = O_1 \cdot F0s^2 + F1s^1 \cdot \overline{O_2} = 0$

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow O_2 := 0 \ (F0s^2 = a)$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \quad \Longrightarrow O_1 := 0$
- $prioS_{Cc_1} = O_1 \cdot F0s^2 + F1s^1 \cdot \overline{O_2} = 0 \begin{cases} prioS^1_{Cc_1} = O_1 \cdot F0s^2 \\ prioS^2_{Cc_1} = F1s^1 \cdot \overline{O_2} \end{cases}$

Résolution des contraintes combinées

Priorité structurelle

Priorité structurelle (Déf. 26 p111)

- Permet de résoudre une contrainte combinée où 1 seule solution est disponible
- Prise en compte des variables commandables forcées par des contraintes simples (fonctions de forçage)

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow O_2 := 0 \ (F0s^2 = a)$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \quad \Longrightarrow O_1 := 0$
- $prioS_{Cc_1} = O_1 \cdot F0s^2 + F1s^1 \cdot \overline{O_2} = 0 \begin{cases} prioS^1_{Cc_1} = O_1 \cdot F0s^2 \\ prioS^2_{Cc_1} = F1s^1 \cdot \overline{O_2} \end{cases}$
- $Cc_2 = a \cdot O_1 \cdot O_2 \cdot \overline{O_3}$
- $prioS_{Cc_2} = a \cdot (O_1 \cdot F1s^2 \cdot F0s^3 + F1s^1 \cdot O_2 \cdot F0s^3 + F1s^1 \cdot F1s^2 \cdot \overline{O_3}) = 0$

Résolution des contraintes combinées

Priorité structurelle

Formalisation (Déf. 26 p111)

Soit une contrainte combinée : $Cc = \prod(\overset{*}{Y}^{Cc}) \cdot \prod(\overset{*}{O}^{Cc})$

Soit $Fs^j = \begin{cases} F0s^j \text{ si } \overline{o_j} \in Cc \\ F1s^j \text{ si } o_j \in Cc \\ 0 \text{ sinon} \end{cases}$ et $\overset{*}{o}_k = \begin{cases} o_k \text{ si } o_k \in Cc \\ \overline{o_k} \text{ si } \overline{o_k} \in Cc \\ 0 \text{ sinon} \end{cases}$

Alors la forme générale des monômes constituant la priorité structurelle de Cc pour o_k est :

$$prioS^k_{Cc} = \prod(\overset{*}{Y}^{Cc}) \cdot \prod_{j=1}^{dim(\overset{*}{O}), j \neq k} (Fs^j) \cdot \overset{*}{o}_k$$

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow F0s^2 = a$
- $Cc_1 = O_1 \cdot \overline{O_2}$

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow F0s^2 = a$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \Longrightarrow \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 \\ O_1 := 0 & \text{et} \quad O_2 := O_2 \\ O_1 := O_1 & \text{et} \quad O_2 := 1 \end{cases}$

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Exemple

- $Cs_1 = a \cdot O_2 \Longrightarrow F0s^2 = a$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \Longrightarrow \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 & : \text{pas de priorité} \\ O_1 := 0 & \text{et} \quad O_2 := O_2 & : \text{priorité à } O_2 \\ O_1 := O_1 & \text{et} \quad O_2 := 1 & : \text{priorité à } O_1 \end{cases}$

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow F0s^2 = a$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \Longrightarrow \begin{cases} \mathbf{O_1 := 0} & \text{et} \quad \mathbf{O_2 := 1} & \text{: pas de priorité} \\ O_1 := 0 & \text{et} \quad O_2 := O_2 & \text{: priorité à } O_2 \\ O_1 := O_1 & \text{et} \quad O_2 := 1 & \text{: priorité à } O_1 \end{cases} \longleftarrow$
- $prioF^1_{Cc_1} = (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2}) \cdot O_1$
- $prioF^2_{Cc_1} = (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2}) \cdot \overline{O_2}$

Résolution des contraintes combinées

Priorité fonctionnelle

Priorité fonctionnelle (Déf. 27 p113)

- Permet de résoudre une contrainte combinée où plusieurs solutions sont disponibles
- Prise en compte d'un choix de résolution "arbitraire"

Exemple

- $Cs_1 = a \cdot O_2 \quad \Longrightarrow F0s^2 = a$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \implies \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 & : \text{pas de priorité} \\ \mathbf{O_1 := 0} & \text{et} \quad \mathbf{O_2 := O_2} & : \text{priorité à } \mathbf{O_2} \\ O_1 := O_1 & \text{et} \quad O_2 := 1 & : \text{priorité à } O_1 \end{cases} \longleftarrow$
- $prioF^1_{Cc_1} = (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2}) \cdot O_1$
- $prioF^2_{Cc_1} = 0$

Résolution des contraintes combinées

Priorité fonctionnelle

Formalisation (Déf. 27 p113)

Soit la contrainte combinée $Cc = \prod(\overset{*}{Y}^{Cc}) \cdot \prod(\overset{*}{O}^{Cc})$

Soit $Fs^j = \begin{cases} F0s^j \text{ si } \overline{o_j} \in Cc \\ F1s^j \text{ si } o_j \in Cc \\ 0 \text{ sinon} \end{cases}$ et $\overset{*}{o_k} = \begin{cases} o_k \text{ si } o_k \in Cc \\ \overline{o_k} \text{ si } \overline{o_k} \in Cc \\ 0 \text{ sinon} \end{cases}$

Pour toute priorité fonctionnelle telle que "$Cc = 1 \implies$ forcer $\overset{*}{o_k}$", un monôme logique est défini :

$$prioF^k_{Cc} = \prod(\overset{*}{Y}^{Cc}) \cdot \prod(\overset{*}{G}^{Cc}) \cdot \prod_{j=1}^{dim(\overset{*}{O}),j\neq k} (\overline{Fs^j}) \cdot \overset{*}{o_k}$$

- $\prod(\overset{*}{Y}^{Cc}) \cdot \prod(\overset{*}{G}^{Cc})$: la contrainte Cc est-elle violée par le vecteur à tester (G) ?
- $\prod_{j=1}^{dim(\overset{*}{O}),j\neq k} (\overline{Fs^j})$: variables commandables de Cc forcées par des contraintes simples ?

Remarque : si aucune priorité fonctionnelle ne concerne $\overset{*}{o_k}$ pour la contrainte Cc, alors $prioF^k_{Cc} = 0$.

Résolution des contraintes combinées

Fonctions de forçage combinées

Fonctions de forçage combinées (Déf. 28 p114)

- Polynômes logiques composés uniquement de variables non-commandables ;
- Permet de savoir si une variable commandable (O_k) doit être forcée à 0 ($F0c^k$) ou à 1 ($F1c^k$) pour résoudre une ou plusieurs contraintes combinées ;
- Prise en compte des fonctions de forçage simples ;
- Construits à partir des parties non-commandables des priorités ;

Résolution des contraintes combinées

Fonctions de forçage combinées

Fonctions de forçage combinées (Déf. 28 p114)

- Polynômes logiques composés uniquement de variables non-commandables ;
- Permet de savoir si une variable commandable (O_k) doit être forcée à 0 ($F0c^k$) ou à 1 ($F1c^k$) pour résoudre une ou plusieurs contraintes combinées ;
- Prise en compte des fonctions de forçage simples ;
- Construits à partir des parties non-commandables des priorités ;

Exemple

- $Cs_1 = a \cdot O_2$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \implies \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 & : \text{pas de priorité} \\ \mathbf{O_1 := 0} & \text{et} \quad \mathbf{O_2 := O_2} & : \text{priorité à } \mathbf{O_2} \\ O_1 := O_1 & \text{et} \quad O_2 := 1 & : \text{priorité à } O_1 \end{cases} \longleftarrow$
- $prioS^1_{Cc_1} = O_1 \cdot F0s^2$
- $prioS^2_{Cc_1} = F1s^1 \cdot \overline{O_2}$
- $prioF^1_{Cc_1} = (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2}) \cdot O_1$
- $prioF^2_{Cc_1} = 0$

Résolution des contraintes combinées

Fonctions de forçage combinées

Fonctions de forçage combinées (Déf. 28 p114)

- Polynômes logiques composés uniquement de variables non-commandables ;
- Permet de savoir si une variable commandable (O_k) doit être forcée à 0 ($F0c^k$) ou à 1 ($F1c^k$) pour résoudre une ou plusieurs contraintes combinées ;
- Prise en compte des fonctions de forçage simples ;
- Construits à partir des parties non-commandables des priorités ;

Exemple

- $Cs_1 = a \cdot O_2$
- $Cc_1 = O_1 \cdot \overline{O_2} = 1 \implies \begin{cases} O_1 := 0 & \text{et} \quad O_2 := 1 & : \text{pas de priorité} \\ \mathbf{O_1 := 0} & \text{et} \quad \mathbf{O_2 := O_2} & : \text{priorité à } \mathbf{O_2} \\ O_1 := O_1 & \text{et} \quad O_2 := 1 & : \text{priorité à } O_1 \end{cases} \longleftarrow$

- $prioS^1_{Cc_1} = O_1 \cdot F0s^2$
- $prioS^2_{Cc_1} = F1s^1 \cdot \overline{O_2}$
- $prioF^1_{Cc_1} = (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2}) \cdot O_1$
- $prioF^2_{Cc_1} = 0$

- $F0c^1 = F0s^2 + (g_1 \cdot \overline{g_2}) \cdot (\overline{F1s^1} \cdot \overline{F0s^2})$
- $F1c^1 = 0$
- $F0c^2 = 0$
- $F1c^2 = F1s^1$

Résolution des contraintes combinées

Fonctions de forçage combinées

Formalisation (Déf. 28 p115)

La fonction de forçage à 0 combinée de $\overset{*}{o_k}$ est définie comme : la somme logique des parties non-commandables des monômes traduisant les priorités structurelles et fonctionnelles concernant o_k.

$$\forall Cc \underline{|} o_k \in Cc, F0c^k = \sum_{Cc} (Y^{prioS^k_{Cc}} + Y^{prioF^k_{Cc}})$$

La fonction de forçage à 1 combinée de $\overset{*}{o_k}$ est définie comme : la somme logique des parties non-commandables des monômes traduisant les priorités structurelles et fonctionnelles concernant $\overline{o_k}$.

$$\forall Cc \underline{|} \overline{o_k} \in Cc, F1c^k = \sum_{Cc} (Y^{prioS^k_{Cc}} + Y^{prioF^k_{Cc}})$$

Utilisation et problématique

Exemple 1

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}}$ (O_2 est forcé)
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3}$ (O_2 est forcé)

Utilisation et problématique

Exemple 1

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}}$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3} \quad \Longrightarrow O_2 = 0$

Exemple 1

$ab|O_1 O_2 O_3$

t_0 $00|110 => Cc_2 = 1 => O_2 := 0$

Utilisation et problématique

Exemple 1

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}} \quad \Longrightarrow O_2 = 1$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3}$

Exemple 1

$ab|O_1 O_2 O_3$

t_0 $00|110 => Cc_2 = 1 => O_2 := 0$

t_1 $00|100 => Cc_1 = 1 => O_2 := 1$

Utilisation et problématique

Exemple 1

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}}$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3} \quad \Longrightarrow O_2 = 0$

Exemple 1

$ab|O_1 O_2 O_3$

t_0 $00|110 => Cc_2 = 1 => O_2 := 0$

t_1 $00|100 => Cc_1 = 1 => O_2 := 1$

t_2 $00|110 => Cc_2 = 1 => O_2 := 0$

etc...

Utilisation et problématique

Exemple 1

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}}$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3} \quad \Longrightarrow O_2 = 0$

Exemple 2

- $Y = \{a, b\}$, $O = \{O_1, O_2, O_3\}$ et $G = \{g_1, g_2, g_3\}$
- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = \mathbf{O_1} \cdot \overline{O_2}$ (O_1 est forcé)
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3}$ (O_2 est forcé)

Exemple 1

$ab|O_1 O_2 O_3$

t_0 $00|110 => Cc_2 = 1 => O_2 := 0$

t_1 $00|100 => Cc_1 = 1 => O_2 := 1$

t_2 $00|110 => Cc_2 = 1 => O_2 := 0$

etc...

Exemple 2

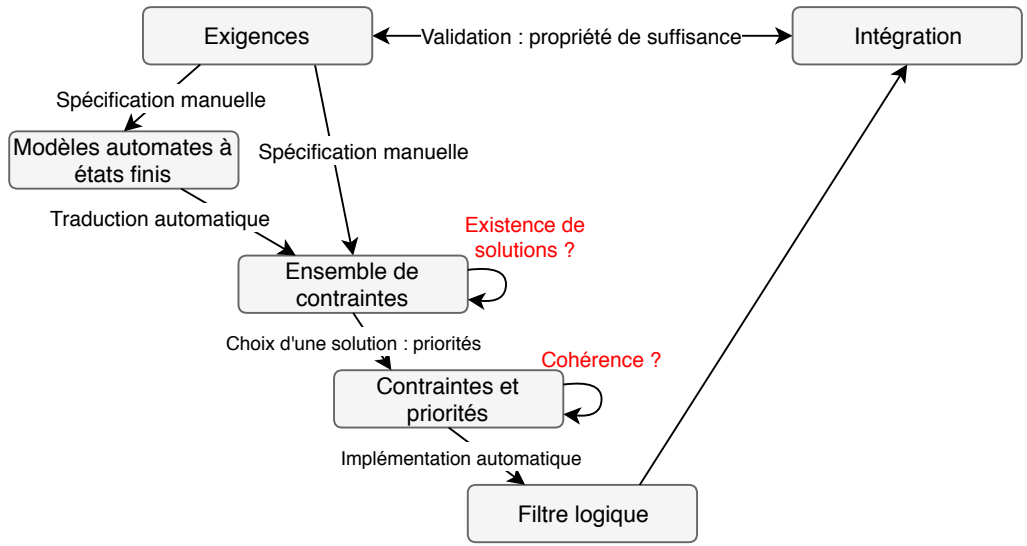
$ab|O_1 O_2 O_3$

t_0 $00|110 => Cc_2 = 1 => O_2 := 0$

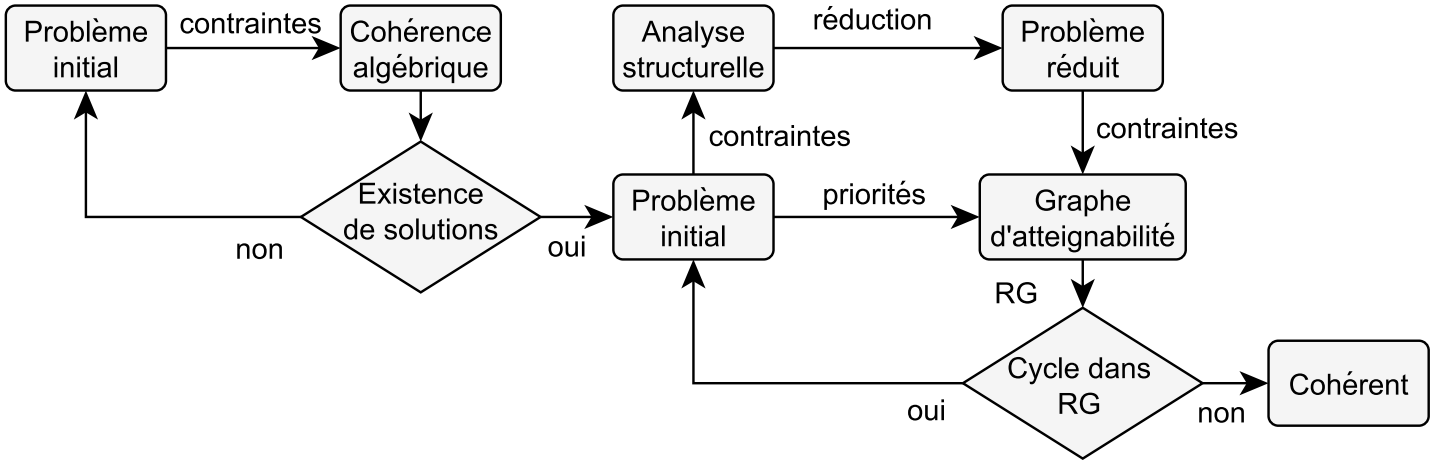
t_1 $00|100 => Cc_1 = 1 => O_1 := 0$

t_2 $00|000 =>$ résolu

Table des matières



Approche proposée



Approche proposée (Pichard et al, 2017b)

1- Existence de solutions
   - Basé uniquement sur les contraintes
   - Utilisation de la synthèse algébrique (Hietter, 2009)

2- Analyse structurelle
   - Basé uniquement sur les contraintes
   - Permet de réduire la taille du problème

3- Graphe d'atteignabilité
   - Basé sur contraintes et priorités
   - Chaque état est une affectation totale des variables
   - Chaque arc représente un pas de résolution

4- Condition de cohérence : existe-t-il un cycle dans le graphe ?

Approche proposée

Analyse structurelle

Vérification de l'existence

- Utilisation de l'outil logiciel BESS dédié à la synthèse algébrique (Hietter, 2009)

TABLE – Problème initial

$Cs_1 = a \cdot O_2$	$Cs_2 = \overline{a} \cdot b \cdot O_1$
$Cc_1 = O_1 \cdot \overline{O_2}$	$Cc_2 = O_2 \cdot \overline{O_3}$

Approche proposée

Analyse structurelle

Vérification de l'existence

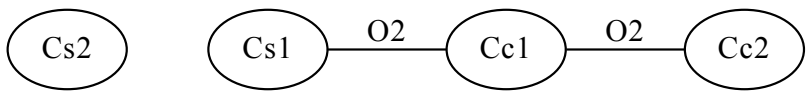
- Utilisation de l'outil logiciel BESS dédié à la synthèse algébrique (Hietter, 2009)

Analyse structurelle

- Construction d'un graphe non-orienté (Déf. 33 et 34 p123)
- Représente les risques de violation lors de la résolution d'une contrainte

TABLE – Problème initial

$Cs_1 = a \cdot O_2$	$Cs_2 = \overline{a} \cdot b \cdot O_1$
$Cc_1 = O_1 \cdot \overline{O_2}$	$Cc_2 = O_2 \cdot \overline{O_3}$



Approche proposée

Analyse structurelle

Vérification de l'existence

- Utilisation de l'outil logiciel BESS dédié à la synthèse algébrique (Hietter, 2009)

Analyse structurelle

- Construction d'un graphe non-orienté (Déf. 33 et 34 p123)
- Représente les risques de violation lors de la résolution d'une contrainte

Réduction du problème

- Suppression des contraintes isolées

TABLE – Problème initial

$Cs_1 = a \cdot O_2$	$Cs_2 = \overline{a} \cdot b \cdot O_1$
$Cc_1 = O_1 \cdot \overline{O_2}$	$Cc_2 = O_2 \cdot \overline{O_3}$

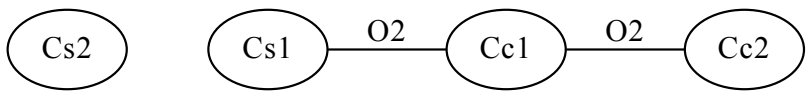


TABLE – Problème réduit

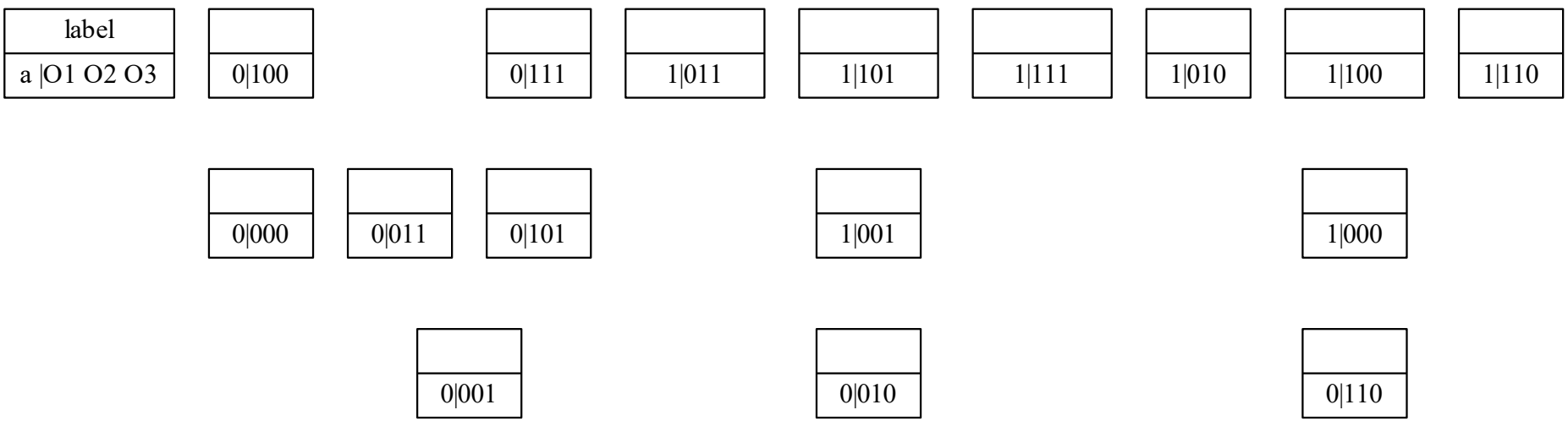
$Cs_1 = a \cdot O_2$	
$Cc_1 = O_1 \cdot \overline{O_2}$	$Cc_2 = O_2 \cdot \overline{O_3}$

Approche proposée

Analyse d'atteignabilité

Construction du graphe d'atteignabilité sur le problème réduit

- Calcul des affectations possibles ;

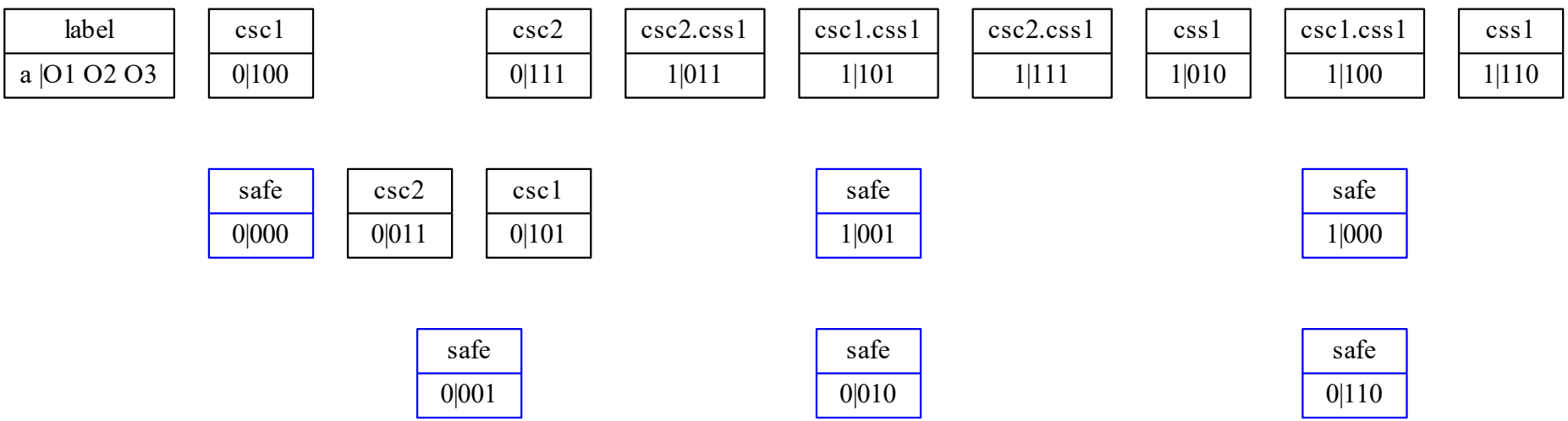


Approche proposée

Analyse d'atteignabilité

Construction du graphe d'atteignabilité sur le problème réduit

- Calcul des affectations possibles ;
- Labellisation des affectations avec les contraintes violées ;

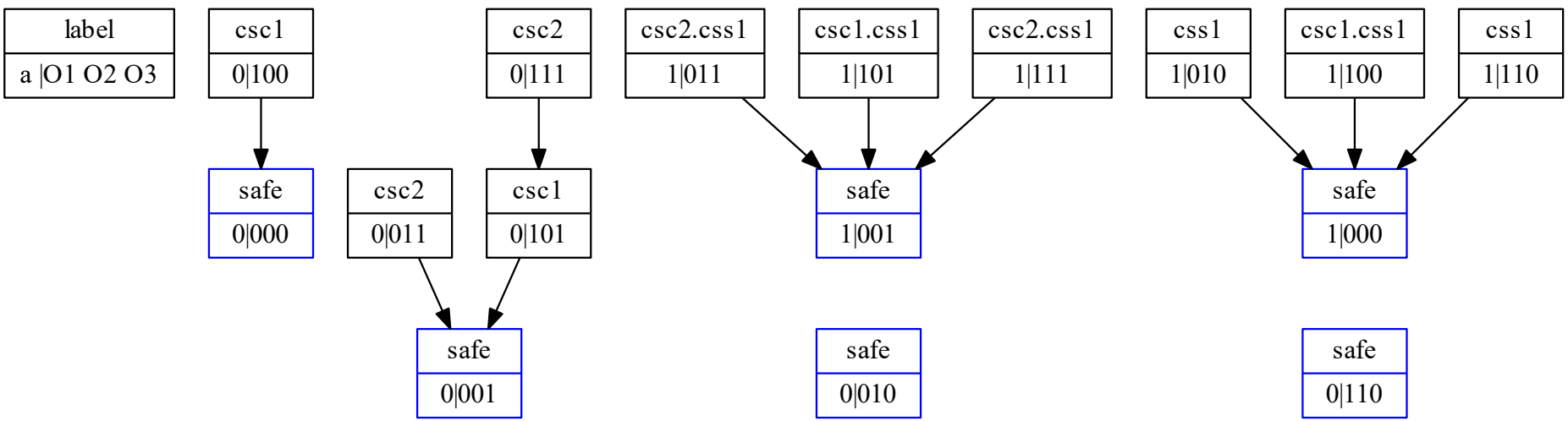


Approche proposée

Analyse d'atteignabilité

Construction du graphe d'atteignabilité sur le problème réduit

- Calcul des affectations possibles ;
- Labellisation des affectations avec les contraintes violées ;
- Création des arcs : résolution et priorités ;

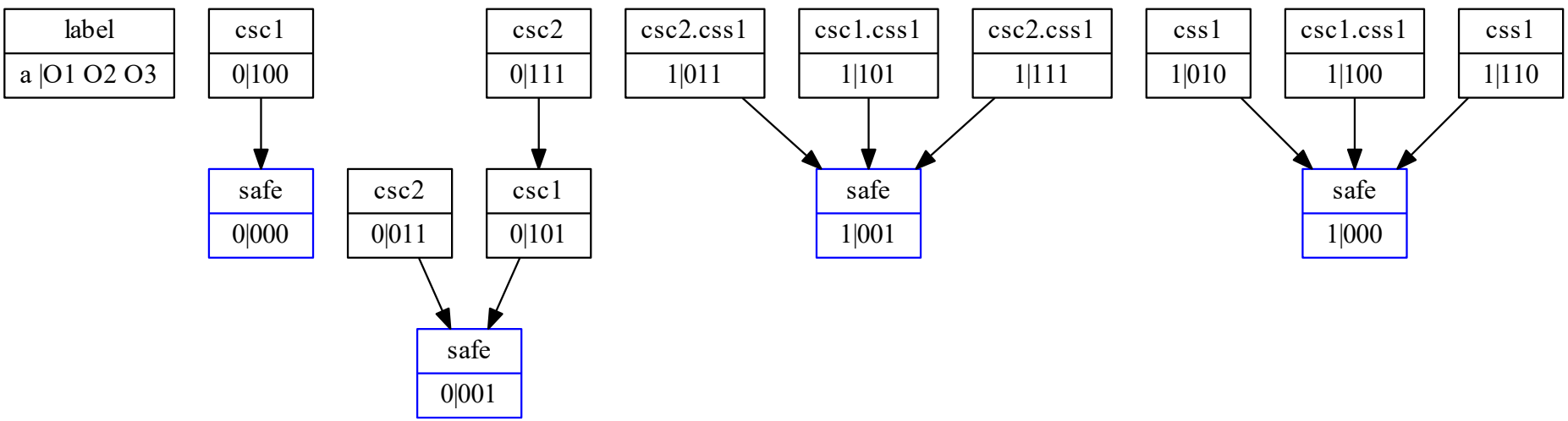


Approche proposée

Analyse d'atteignabilité

Construction du graphe d'atteignabilité sur le problème réduit

- Calcul des affectations possibles ;
- Labellisation des affectations avec les contraintes violées ;
- Création des arcs : résolution et priorités ;
- Réduction du graphe ;

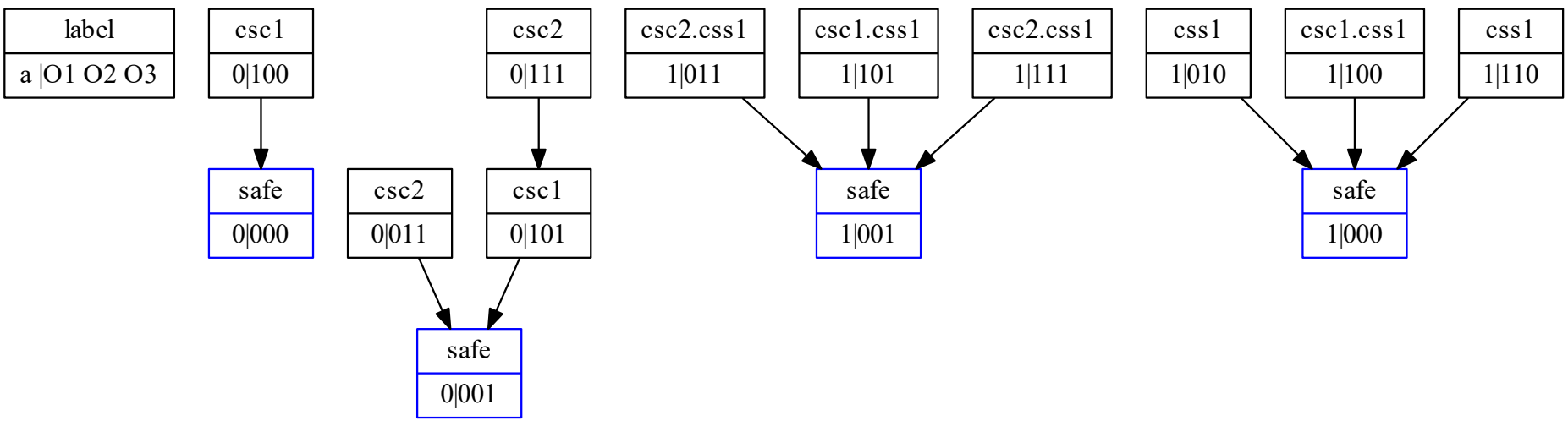


Approche proposée

Analyse d'atteignabilité

Construction du graphe d'atteignabilité sur le problème réduit

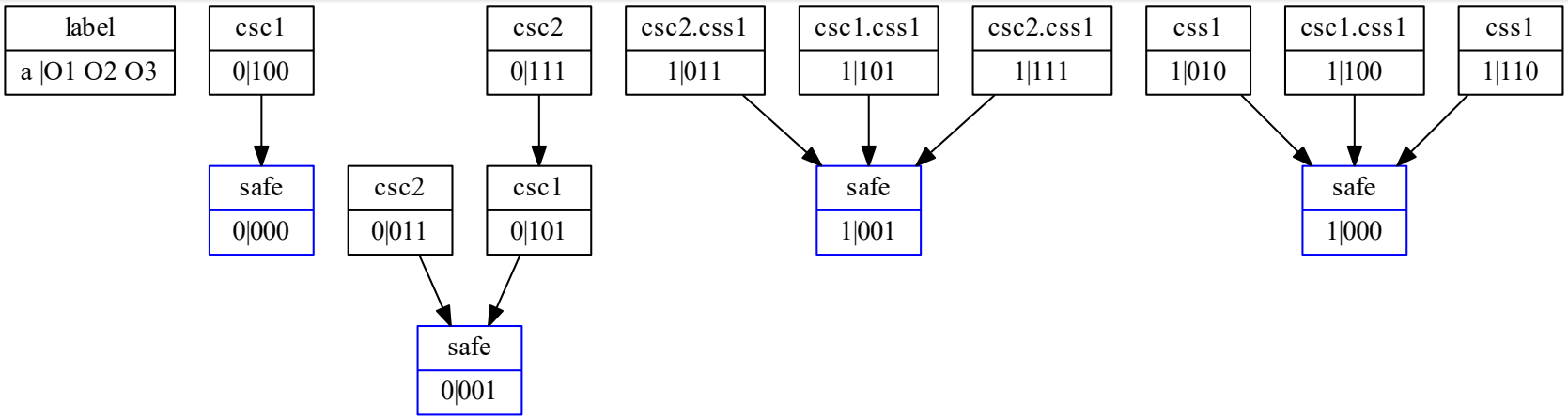
- Calcul des affectations possibles ;
- Labellisation des affectations avec les contraintes violées ;
- Création des arcs : résolution et priorités ;
- Réduction du graphe ;
- Vérification de la cohérence : non-existence d'un cycle ;



Approche proposée

Exemple cohérent

- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = \mathbf{O_1} \cdot \overline{O_2}$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3}$



Exemple non cohérent

- $Cs_1 = a \cdot O_2$
- $Cs_2 = \overline{a} \cdot b \cdot O_1$
- $Cc_1 = O_1 \cdot \overline{\mathbf{O_2}}$
- $Cc_2 = \mathbf{O_2} \cdot \overline{O_3}$

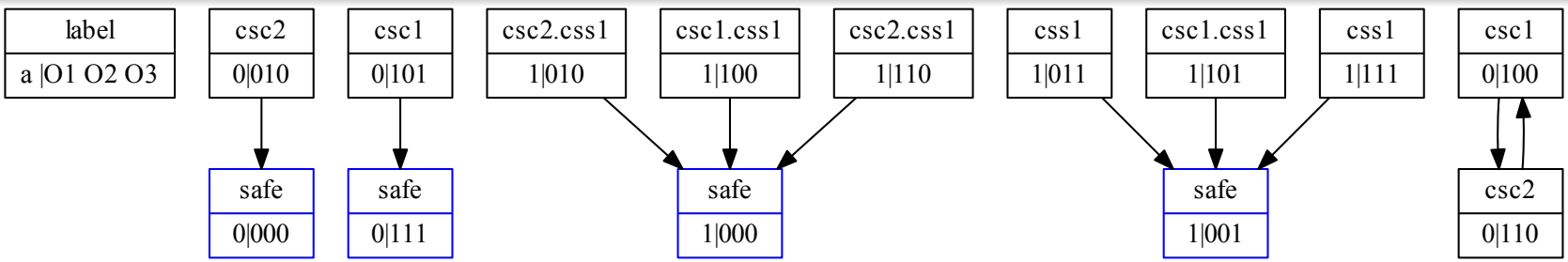


Table des matières

Introduction

État de l'art

Formalisation

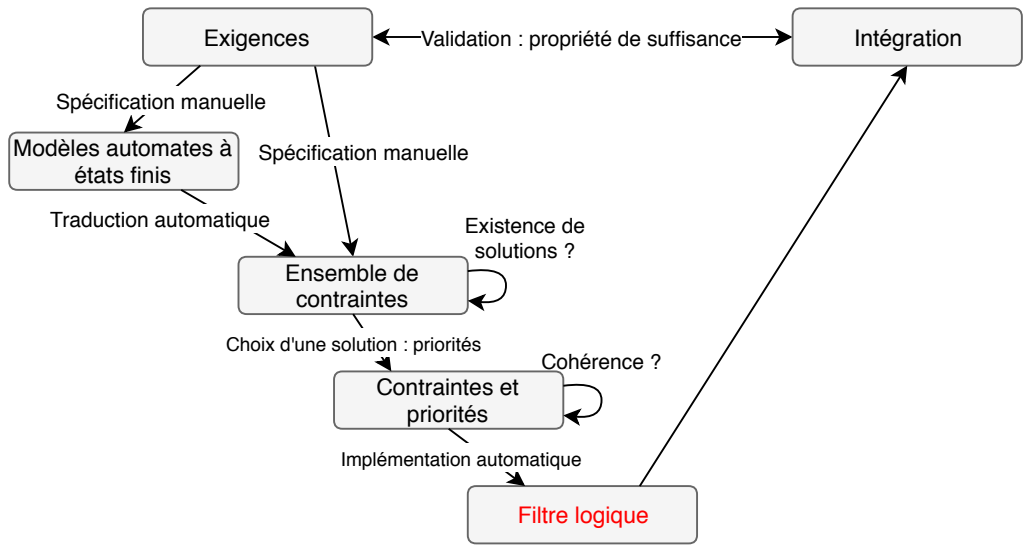
Cohérence

Implémentation

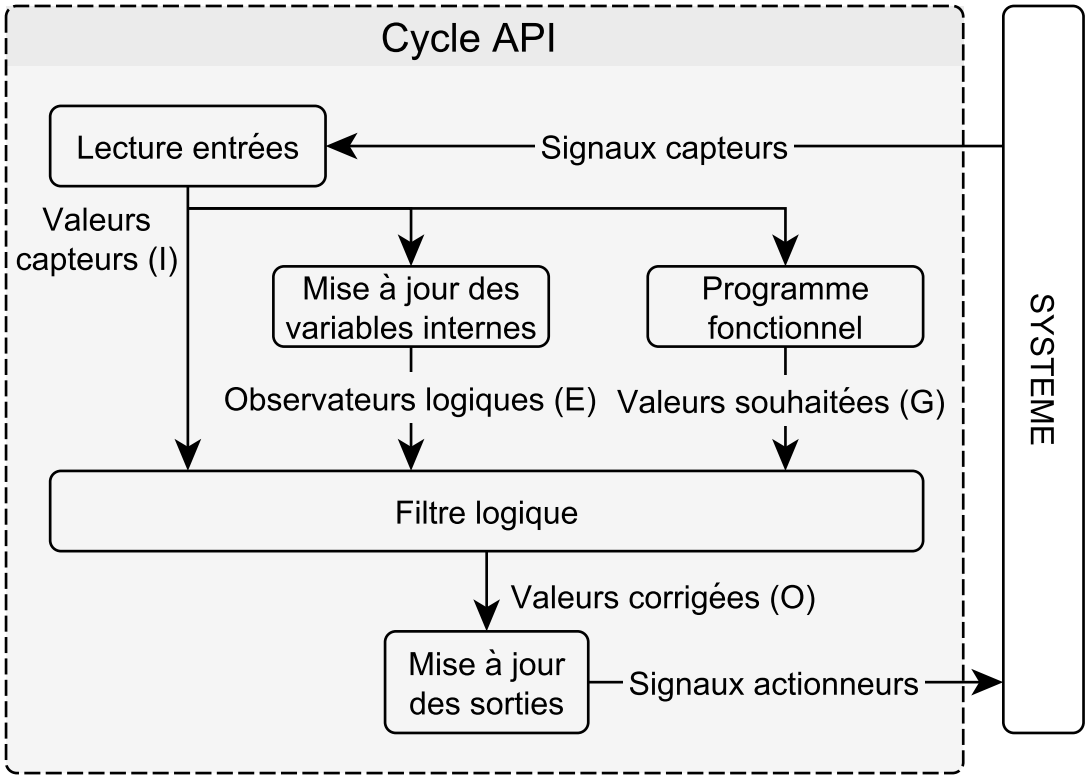
- Objectifs
- Algorithme itératif
- Problème et solveur SAT
  - Solveur SAT pour le filtre logique
  - Recherche locale par distance de Hamming
  - Recherche locale par solveur SAT
- Comparaison

Logiciel SEDMA

Conclusion et perspectives



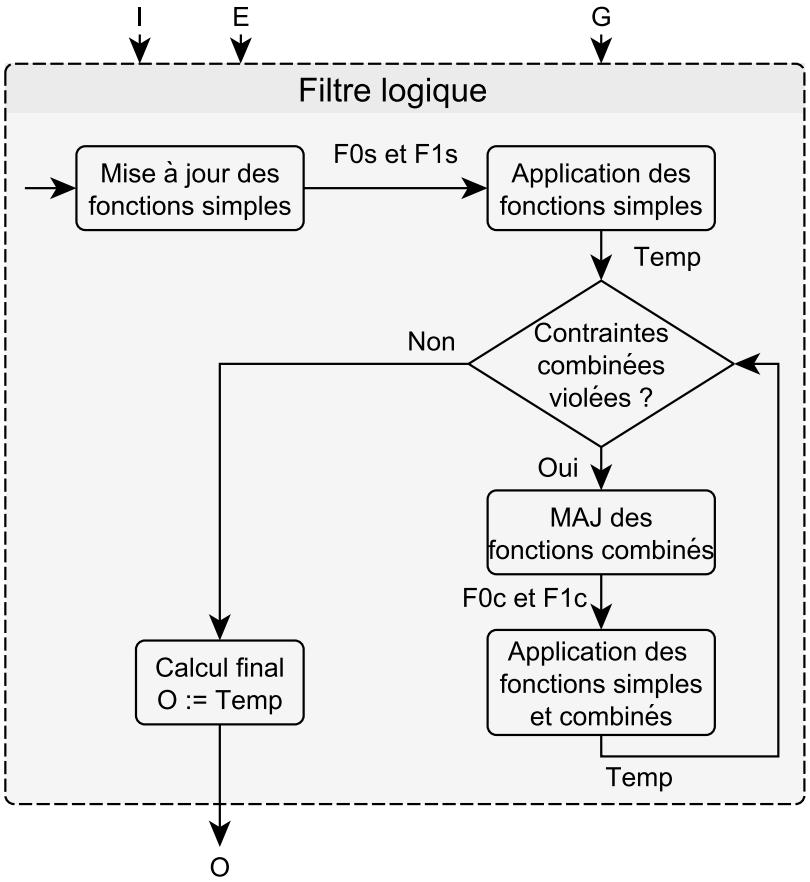
Objectifs



Objectifs

- Être capable de garantir que les valeurs corrigées O, basées sur G, respectent les contraintes
- Algorithmes implémentables dans un API (langage ST)
- Aucune hypothèse sur le programme fonctionnel en amont du filtre

Algorithme itératif



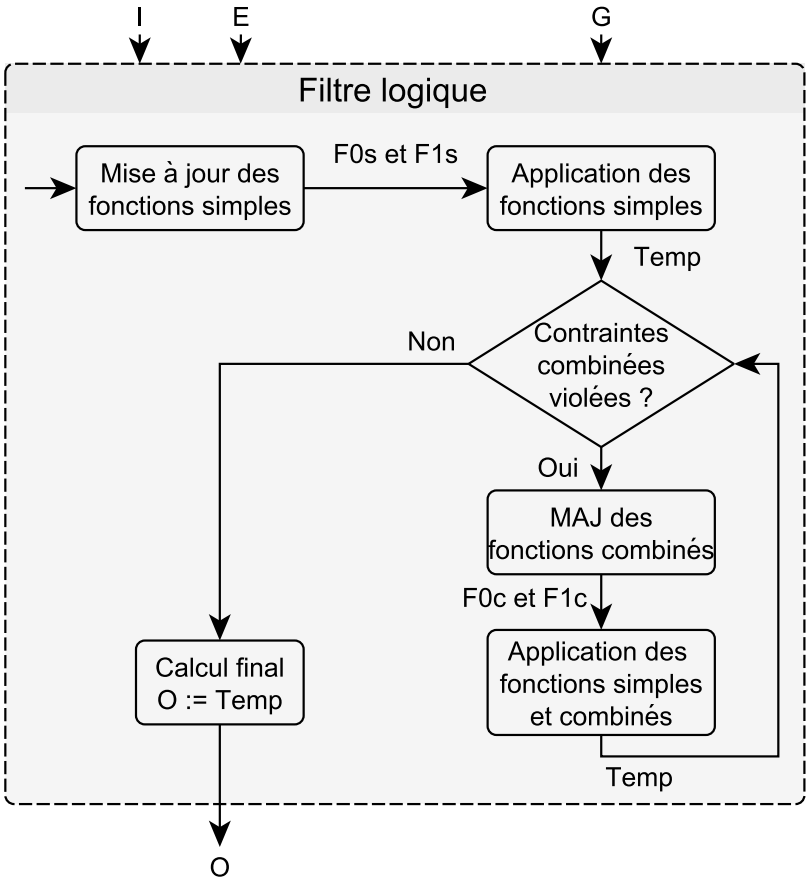
Hypothèses

- Contraintes cohérentes algébriquement
- Priorités fonctionnelles cohérentes avec les contraintes

Caractéristiques (Pichard et al., 2018a)

- Basé sur l'algorithme de (Coupat, 2014)
- Utilise les contraintes et les priorités
- Convergence garantie par les hypothèses de cohérence

Algorithme itératif



Hypothèses

- Contraintes cohérentes algébriquement
- ~~Priorités fonctionnelles cohérentes avec les contraintes~~

Caractéristiques (Pichard et al., 2018a)

- Basé sur l'algorithme de (Coupat, 2014)
- Utilise les contraintes ~~et les priorités~~
- Convergence garantie par les hypothèses de cohérence

Problème et solveur SAT

$$\overline{O_1} \cdot O_2 = 0 \qquad \overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$$

XXX

choix
propagation
back-tracking

O_1

O_2

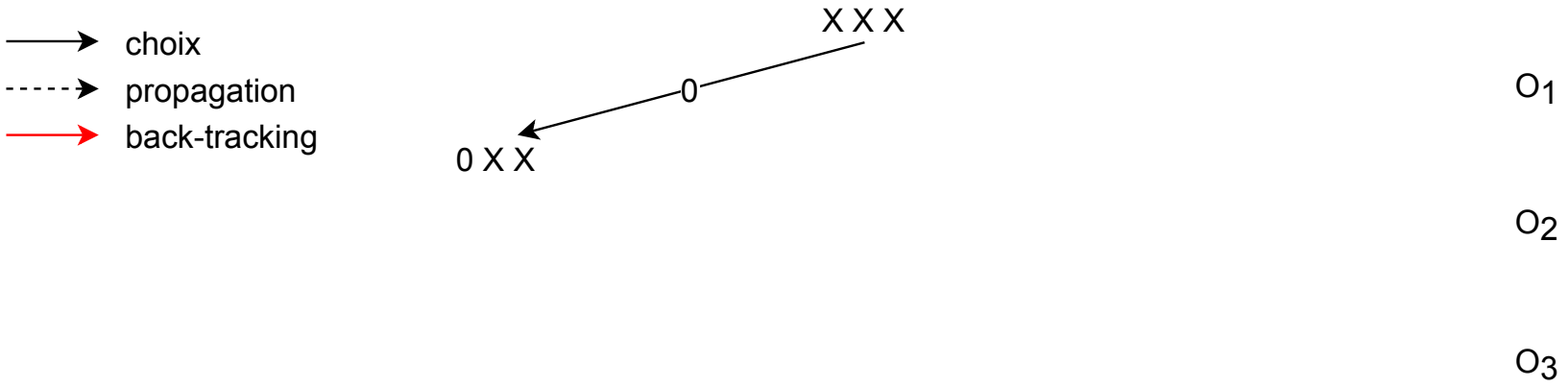
O_3

Problème SAT (SATisfiabilité) (Davis et al., 1962)

- Problème SAT : composé de contraintes portant sur des variables booléennes
- Solveur SAT : permet de trouver une ou plusieurs solutions (affectations des variables) vérifiant l'ensemble des contraintes

Problème et solveur SAT

$\overline{O_1} \cdot O_2 = 0$ $\overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$

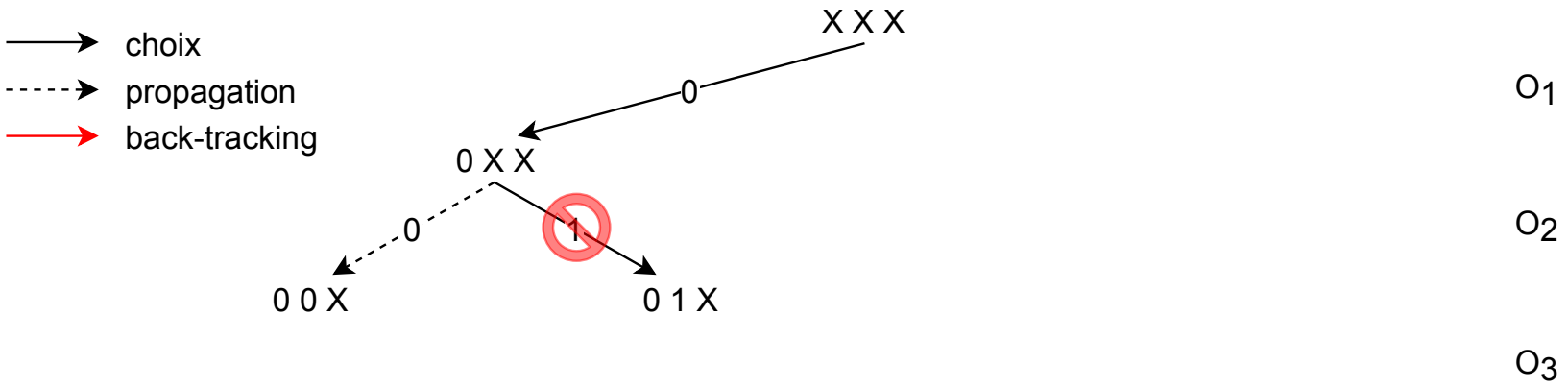


Problème SAT (SATisfiabilité) (Davis et al., 1962)

- Problème SAT : composé de contraintes portant sur des variables booléennes
- Solveur SAT : permet de trouver une ou plusieurs solutions (affectations des variables) vérifiant l'ensemble des contraintes

Problème et solveur SAT

$\overline{O_1} \cdot O_2 = 0$ $\overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$

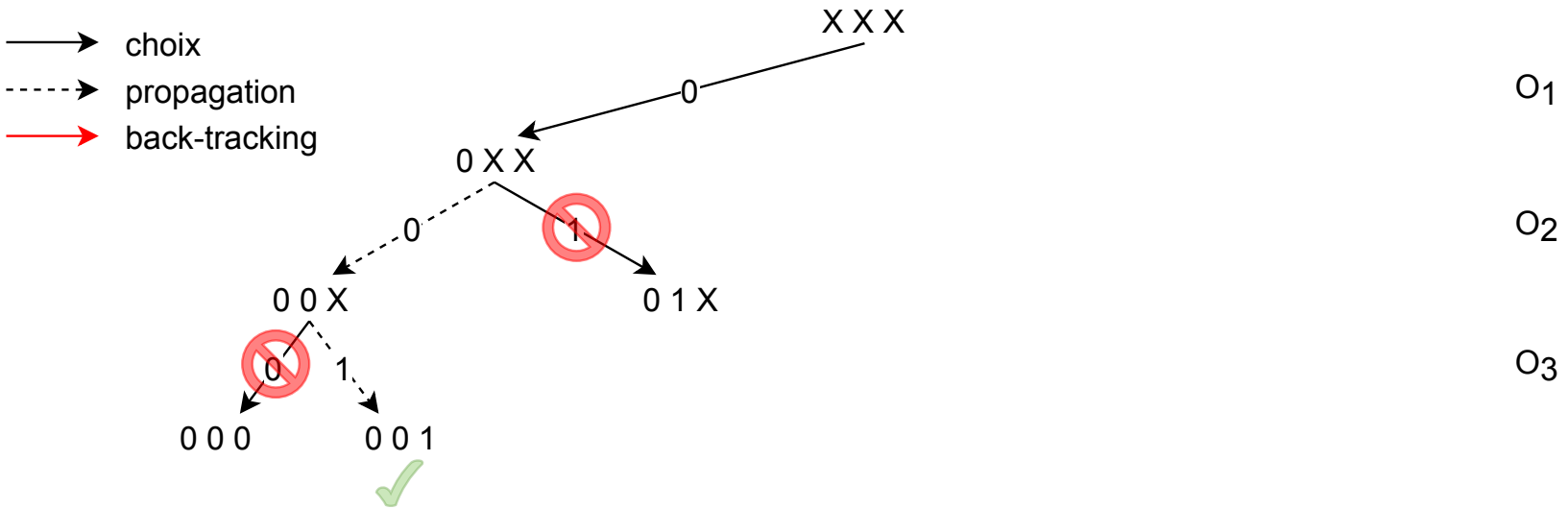


Problème SAT (SATisfiabilité) (Davis et al., 1962)

- Problème SAT : composé de contraintes portant sur des variables booléennes
- Solveur SAT : permet de trouver une ou plusieurs solutions (affectations des variables) vérifiant l'ensemble des contraintes

Problème et solveur SAT

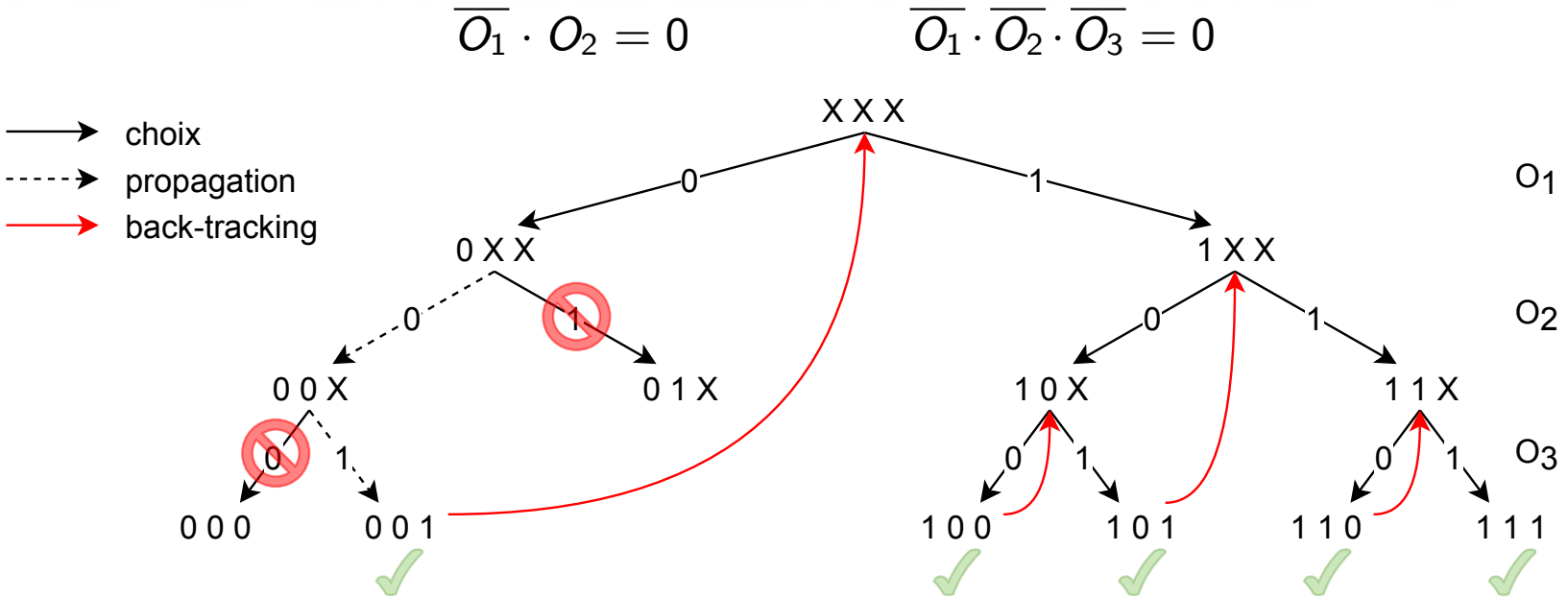
$\overline{O_1} \cdot O_2 = 0$ $\overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$



Problème SAT (SATisfiabilité) (Davis et al., 1962)

- Problème SAT : composé de contraintes portant sur des variables booléennes
- Solveur SAT : permet de trouver une ou plusieurs solutions (affectations des variables) vérifiant l'ensemble des contraintes

Problème et solveur SAT

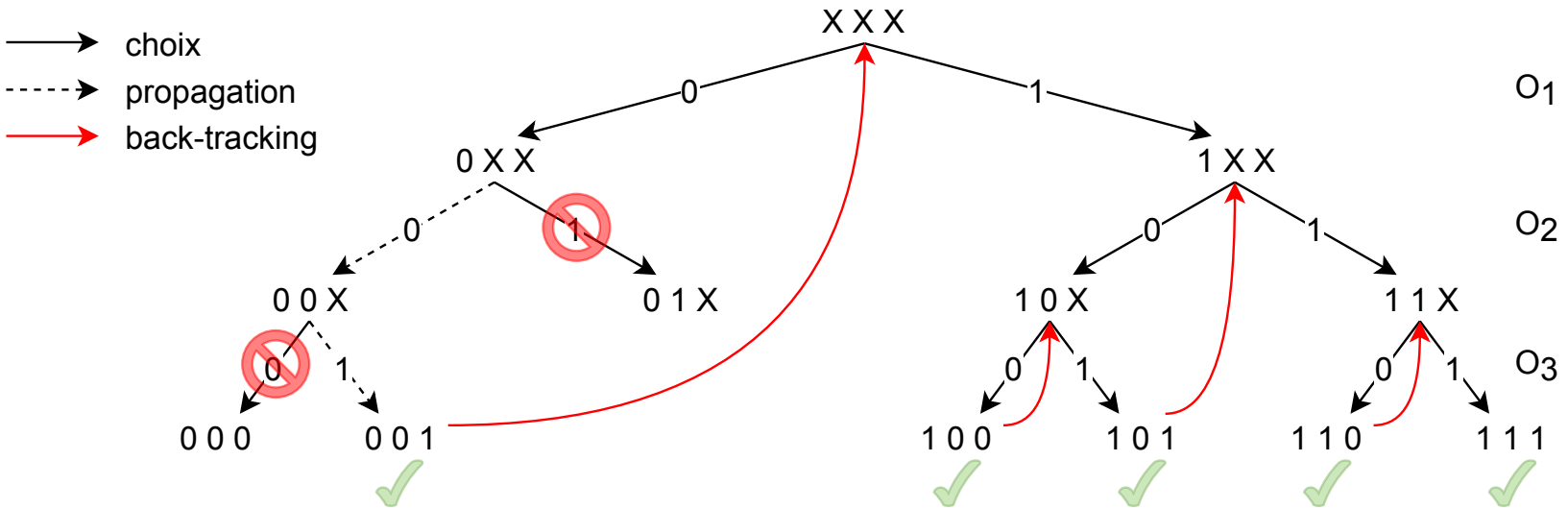


Problème SAT (SATisfiabilité) (Davis et al., 1962)

- Problème SAT : composé de contraintes portant sur des variables booléennes
- Solveur SAT : permet de trouver une ou plusieurs solutions (affectations des variables) vérifiant l'ensemble des contraintes

Solveur SAT pour le filtre logique

$G = 0\ 1\ 0$ $\quad \overline{O_1} \cdot O_2 = 0$ $\quad \overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$



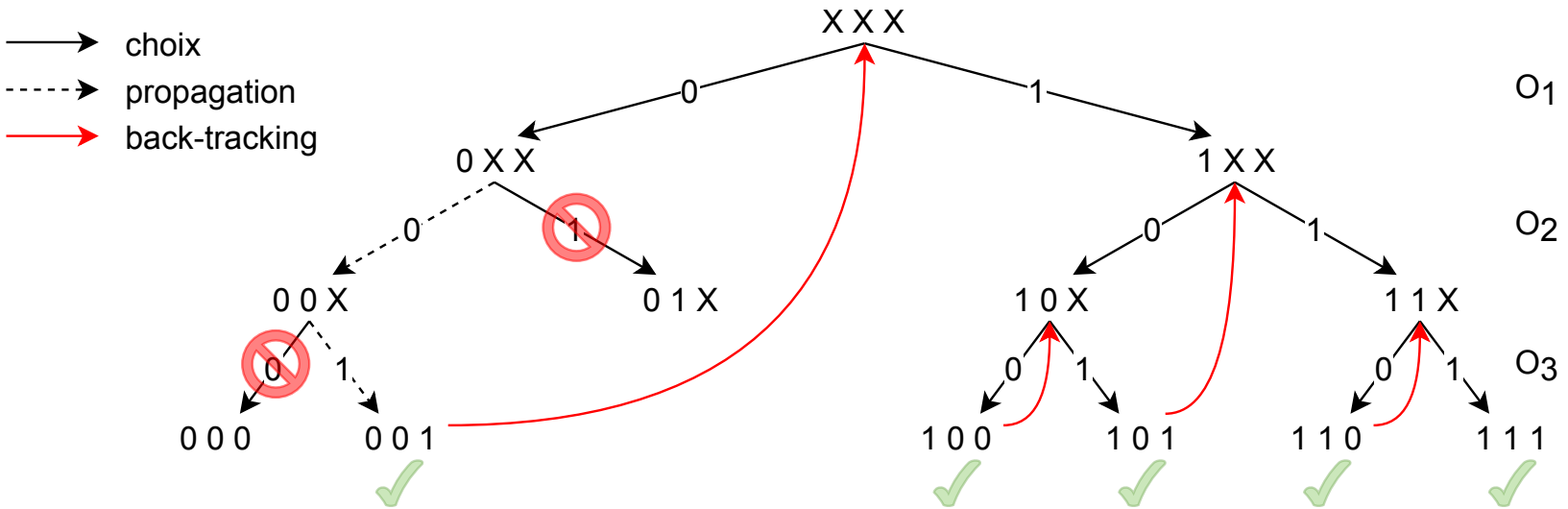
G	010
O	001
	100
	101
	110
	111

Solveur SAT et distance de Hamming (Pichard et al., 2016)

- Preuve de concept utilisant un solveur SAT externe à l'API
- Calcul des différentes solutions par le solveur SAT

Solveur SAT pour le filtre logique

$G = 0\ 1\ 0$ $\quad \overline{O_1} \cdot O_2 = 0$ $\quad \overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$



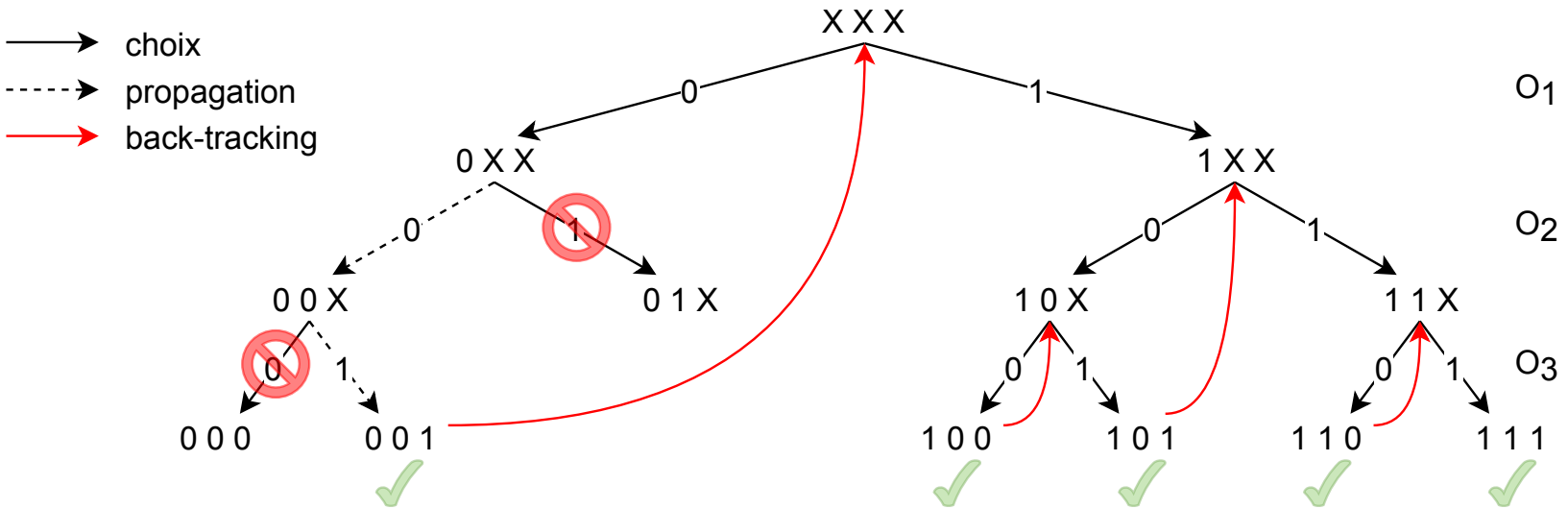
G	010	Hamming
O	001	2
	100	2
	101	3
	110	1
	111	2

Solveur SAT et distance de Hamming (Pichard et al., 2016)

- Preuve de concept utilisant un solveur SAT externe à l'API
- Calcul des différentes solutions par le solveur SAT

Solveur SAT pour le filtre logique

$G = 0\ 1\ 0$ $\quad \overline{O_1} \cdot O_2 = 0$ $\quad \overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$



G	010	Hamming
O	001	2
	100	2
	101	3
	110	1
	111	2

Solveur SAT et distance de Hamming (Pichard et al., 2016)

- Preuve de concept utilisant un solveur SAT externe à l'API
- Calcul des différentes solutions par le solveur SAT
- Choix : distance de Hamming minimale

Recherche locale par distance de Hamming

$$G = 0010$$

$$\text{dist} = 1 \begin{cases} 1010 \\ 0110 \\ 0000 \\ 0011 \end{cases}$$

Recherche locale par distance croissante (Pichard et al., 2018b)

- Calcul et test des vecteurs par distance de Hamming croissante
- Solution : premier vecteur valide minimisant la distance
- Astuce de calcul pour diviser par 2 le temps de recherche

Recherche locale par distance de Hamming

$$G = 0010$$

$$\text{dist} = 1 \begin{cases} 1010 \\ 0110 \\ 0000 \\ 0011 \end{cases}$$

$$\text{dist} = 2 \begin{cases} 1110 \\ 1000 \\ 1011 \\ 0100 \\ 0111 \\ 0001 \end{cases}$$

Recherche locale par distance croissante (Pichard et al., 2018b)

- Calcul et test des vecteurs par distance de Hamming croissante
- Solution : premier vecteur valide minimisant la distance
- Astuce de calcul pour diviser par 2 le temps de recherche

Recherche locale par distance de Hamming

$$G = 0010$$

$$\text{dist} = 1 \begin{cases} 1010 \\ 0110 \\ 0000 \\ 0011 \end{cases}$$

$$\text{dist} = 2 \begin{cases} 1110 \\ 1000 \\ 1011 \\ 0100 \\ 0111 \\ 0001 \end{cases}$$

$$\text{dist} = 3 \begin{cases} 1100 \\ 1111 \\ 1001 \\ 0101 \end{cases}$$

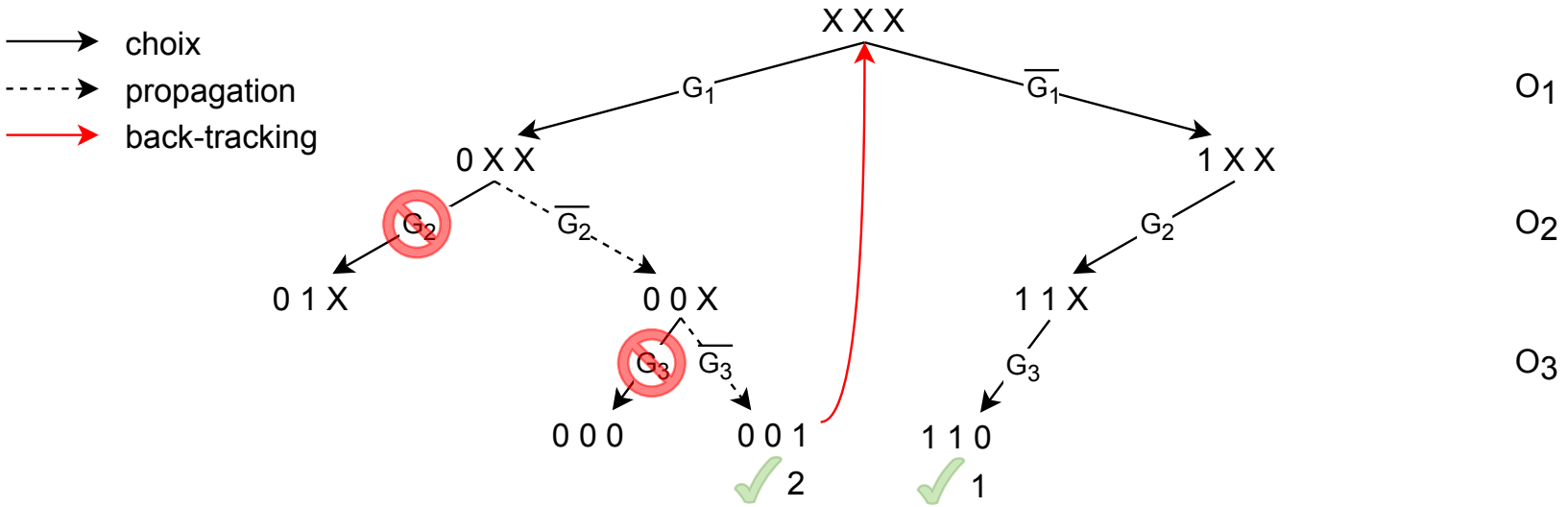
$$\text{dist} = 4 \begin{cases} 1101 \end{cases}$$

Recherche locale par distance croissante (Pichard et al., 2018b)

- Calcul et test des vecteurs par distance de Hamming croissante
- Solution : premier vecteur valide minimisant la distance
- Astuce de calcul pour diviser par 2 le temps de recherche

Recherche locale par solveur SAT

$G = 0\ 1\ 0$ $\quad \overline{O_1} \cdot O_2 = 0$ $\quad \overline{O_1} \cdot \overline{O_2} \cdot \overline{O_3} = 0$



Recherche locale avec back-tracking et distance de Hamming

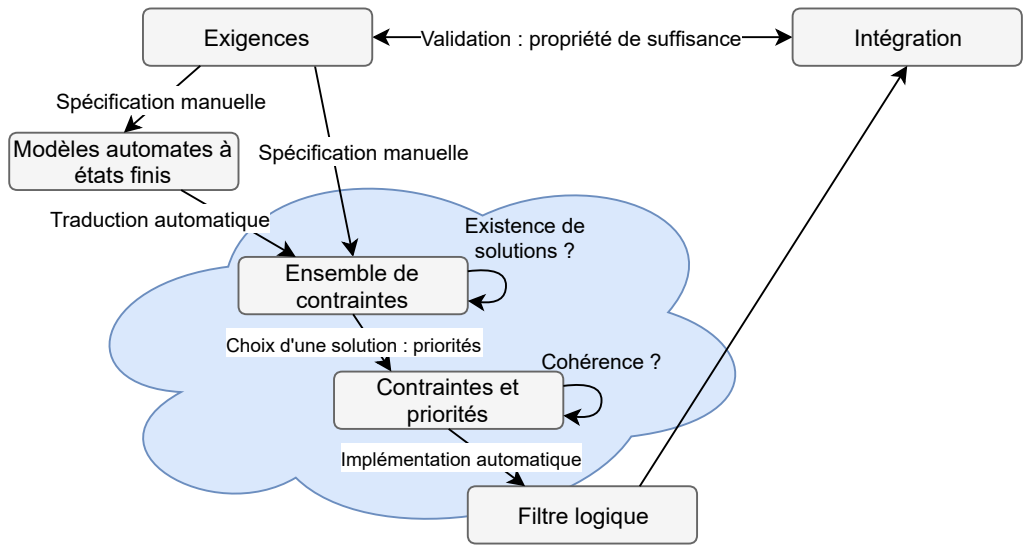
- Utilise les valeurs de G lors des choix d'affectation
- Cherche si possible une solution à distance 1, sinon parcours complet et garde la solution minimisant la distance

Comparaison

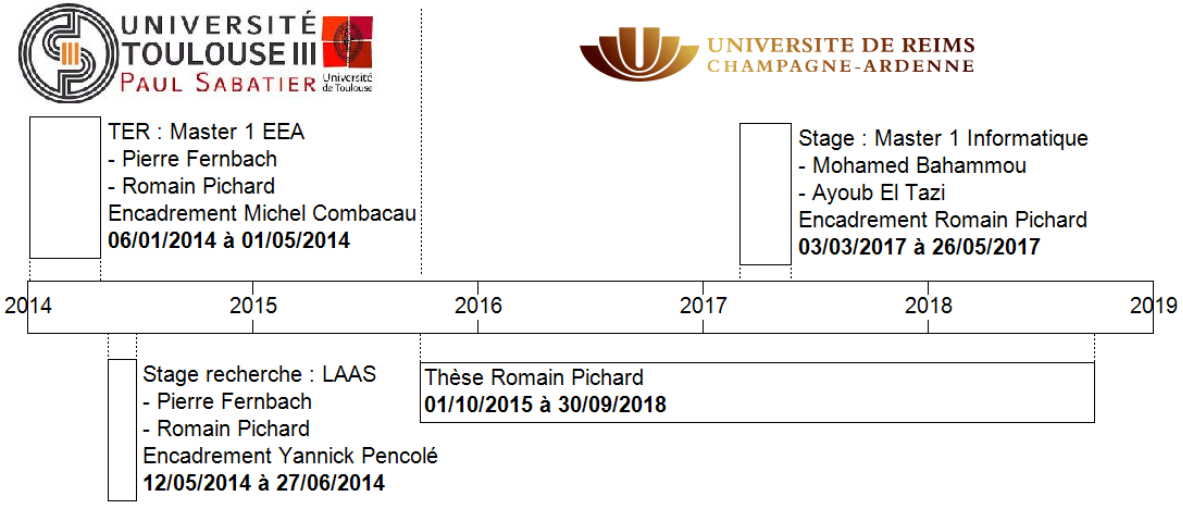
TABLE – Comparaison des algorithmes proposés

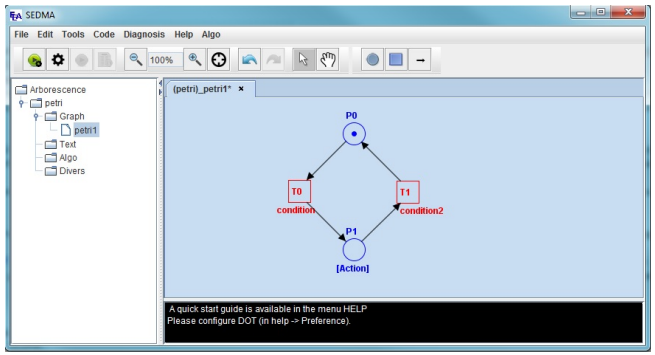
	Alg. itératif	Alg. Hamming	Solveur SAT
Priorités	oui	non	non
Cohérence	globale	algébrique	algébrique
Implémentation	très simple	difficile	très difficile
Maintenabilité	simple	très difficile	très difficile
Espace mémoire	faible	moyen	très grand
Flexibilité	aucune	aucune	critère d'optimisation heuristique de choix
Génération automatique	SEDMA		

Table des matières



Logiciel SEDMA





Développement durant la thèse (Pichard et al., 2017a)

- Restructuration complète du logiciel pour faciliter l'évolution (encadrement de 2 stagiaires)
- Développement des fonctionnalités permettant de supporter l'approche par filtre logique :
  - Saisie textuelle des variables, contraintes et priorités
  - Analyse de la cohérence (construction des graphes et détection de cycles)
  - Génération automatique de l'algorithme itératif en ST
  - Génération automatique de la déclaration des variables pour API Schneider

Application CellFlex

	convoyeur	transfert	import/export	total
capteurs	27	14	17	58
variables internes	10	15	22	47
actionneurs	9	10	13	32
contraintes simples	8	25	51	84
contraintes combinées	0	6	5	11
contraintes structurelles	0	7	12	19
analyse structurelle (ms)	14	97	110	221
analyse cycle (ms)	0	0	525	525

Table des matières

Conclusion

Objectifs

- Généraliser l'approche par filtre logique : formalisme, méthodologie
- Proposer une condition nécessaire et suffisante pour vérifier la cohérence
- Proposer des algorithmes d'implémentation plus flexibles

Contributions

- Proposition d'une approche de conception globale
- Formalisation des définitions et propriétés
- Condition nécessaire et suffisante à la cohérence par analyse de graphe
- Implémentation du filtre sous la forme de solveurs SAT pour API
- Développement d'un outil logiciel pour supporter l'approche de conception

Perspectives

Perspectives

- Explorer la notion de "couverture" des contraintes pour minimiser la taille de l'ensemble de contraintes
- Analyser la cohérence avec des propriétés structurelles (utilisation des RdP ?)
- Appliquer l'approche par filtre logique à d'autres problématiques (reconfiguration, ré-ordonnancement en ligne)
- Explorer d'autres heuristiques d'optimisation pour les solveurs SAT
- Génération automatique des contraintes à partir de modèles standardisés

Merci

Merci de votre attention

Bibliographie

R. Pichard, N. Ben Rabah, V. Carré-Ménétrier et B. Riera : CSP solver for Safe PLC Controller : Application to manufacturing systems. In **IFAC MIM'16**, volume 49 de 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 Troyes, France, 28-30 June 2016, pages 402-407, Troyes, 2016

R. Pichard, M. Combacau, A. Philippot, R. Saddem et B. Riera : SEDMA - un outil pour la Modélisation, l'Analyse et la génération automatique de programme pour les SEDs. In **MSR 2017**, 11ème Colloque sur la Modélisation des Systèmes Réactifs, page 2, Marseille, novembre 2017a

R. Pichard, A. Philippot et B. Riera : Consistency Checking of Safety Constraints for Manufacturing Systems with Graph Analysis. In **20th IFAC WorldCongress**, volume 50 de 20th IFAC World Congress, pages 1193-1198, Toulouse, juillet 2017b

R. Pichard, A. Philippot, R. Saddem et B. Riera : Safety of Manufacturing Systems Controllers by Logical Constraints With Safety Filter. **IEEE Transactions on Control Systems Technology**, pages 1-9, 2018a. ISSN 1063-6536.

R. Pichard, A. Philippot et B. Riera : Safe PLC Controller implementation IEC 61131-3 compliant based on a simple SAT solver : Application to manufacturing systems. In **ICINCO'18**, Porto, juillet 2018b

Bibliographie

C. G. Cassandras et S. Lafortune : Introduction to Discrete Event Systems. Springer Science & Business Media, décembre 2009. ISBN 978-0-387-33332-8.

Brandin, B. A., & Wonham, W. M. (1994). Supervisory control of timed discrete-event systems. IEEE Transactions on Automatic Control, 39(2), 329-342.

Ramadge, P. J., & Wonham, W. M. (1989). The control of discrete event systems. Proceedings of the IEEE, 77(1), 81-98.

Zaytoon, J., & Riera, B. (2017). Synthesis and implementation of logic controllers-a review. Annual reviews in control, 43, 152-168.

Hietter, Y. (2009). Synthèse algébrique de lois de commande pour les systèmes à évènements discrets logiques (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

Bibliographie

Marangé, P. (2008). Synthèse et filtrage robuste de la commande pour des système manufacturiers sûrs de fonctionnement (Doctoral dissertation, Université de Reims-Champagne Ardenne).

Benlorhfar, R., Annebicque, D., Gellot, F., & Riera, B. (2011). Robust filtering of PLC program for automated systems of production. In 18th World Congress of the International Federation of Automatic Control.

Coupat, R. (2014). Méthodologie pour les études d'automatisation et la génération automatique de programmes Automates Programmables Industriels sûrs de fonctionnement. Application aux Équipements d'Alimentation des Lignes Électrifiées (Doctoral dissertation, Université de Reims-Champagne Ardenne).

Riera, B., Philippot, A., Annebicque, D., & Gellot, F. (2015). La commande par contraintes logiques de sécurité : principe, applications et mise en oeuvre. In Modélisation des Systèmes Réactifs (MSR 2015).

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem-proving. Communications of the ACM, 5(7), 394-397.

Algorithme itératif

```
Data: No
Input: I, E, G
Output: O
/* Calcul de F0s^k et F1s^k                          */
for k = 1, ..., No do
    F0s(k) := ... ;
    F1s(k) := ... ;
end
/* Résolution des contraintes simples                */
for k = 1, ..., No do
    Temp(k) := NOT F0s(k) AND G(k) OR F1s(k) ;
    F0c[k] := false ;
    F1c[k] := false ;
end
Flag := true ;
while Flag do
    Cc := updateCc(Temp, I, E) ;
    Flag := estViolée(Cc) ;
    if Flag then
        for k = 1, ..., No do
            F0c(k) := ... ;
            F1c(k) := ... ;
        end
        /* Résolution des contraintes
           combinées                                  */
        for k = 1, ..., No do
            Temp(k) := NOT(F0s(k) OR F0c(k))
              AND Temp(k) OR (F1s(k) OR
              F1c(k)) ;
        end
    end
end
/* Mise à jour des sorties                           */
for k = 1, ..., No do
    O(k) := Temp(k) ;
end
```

Tri de caisse

Description

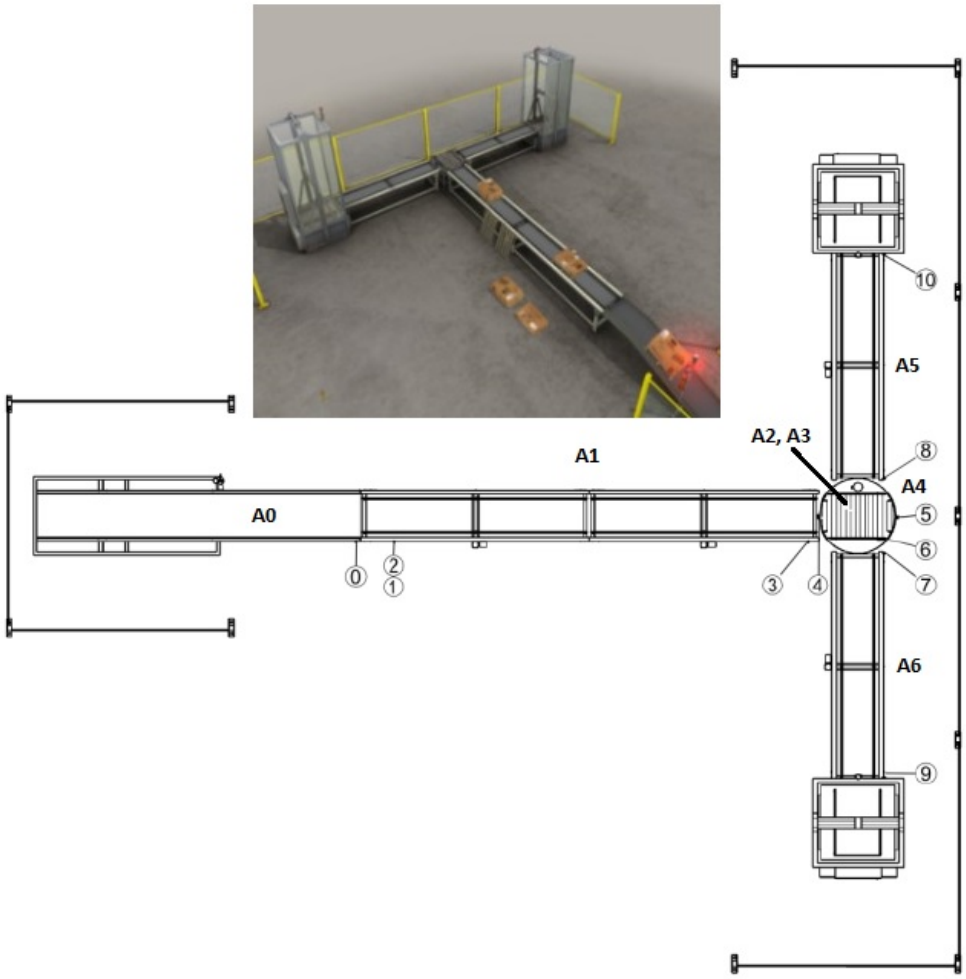
- 11 capteurs : $\{c0, c1, \ldots, c10\}$
- 7 actionneurs : $\{A0, A1, \ldots, A6\}$
- 5 observateurs logiques : $\{2p, p36, p67, p79, p810\}$

Cahiers des charges

- Trier les caisses en fonctions de leur hauteur
- Empêcher les chutes et les collisions

Objectif

- Garantir la sécurité quel que soit le programme API en amont du filtre



Tri de caisse

Contraintes simples

- $CSs_1 : 2p \cdot A0$
- $CSs_2 : c3 \cdot \overline{c4} \cdot A1$
- $CSs_3 : c3 \cdot c4 \cdot c6 \cdot A1$
- $CSs_4 : c3 \cdot p36 \cdot A1$
- $CSs_5 : c4 \cdot c6 \cdot A2$
- $CSs_6 : \overline{c4} \cdot \overline{c5} \cdot A2$
- $CSs_7 : c5 \cdot c7 \cdot A2$
- $CSs_8 : \overline{c5} \cdot A3$
- $CSs_9 : c5 \cdot c6 \cdot \overline{A4}$
- $CSs_{10} : c5 \cdot c8 \cdot \overline{A4}$
- $CSs_{11} : c5 \cdot c7 \cdot \overline{A4}$
- $CSs_{12} : c5 \cdot p67 \cdot \overline{A4}$
- $CSs_{13} : c4 \cdot c9 \cdot A4$
- $CSs_{14} : c4 \cdot \overline{c6} \cdot A4$
- $CSs_{15} : c4 \cdot p79 \cdot A4$
- $CSs_{16} : c4 \cdot p810 \cdot A4$
- $CSs_{17} : c4 \cdot c10 \cdot A4$

Contraintes combinées

- $CSc_1 : c0 \cdot A0 \cdot \overline{A1}$
- $CSc_2 : c3 \cdot c4 \cdot A1 \cdot \overline{A2}$
- $CSc_3 : c5 \cdot c8 \cdot A2 \cdot \overline{A5}$
- $CSc_4 : c5 \cdot c7 \cdot A3 \cdot \overline{A6}$
- $CSc_5 : A2 \cdot A3$

Contrainte structurelle

- $Cstruct_1 : c4 \cdot c5$