Vérification Formelle dans le Modèle Polyédrique

Katell Morin-Allory

Thèse encadrée par David Cachera et Patrice Quinton

Contexte

Thèse effectuée dans 2 équipes de l'IRISA : R2D2 et Lande

- R2D2(Cosi) :
  - conception système sur silicium
  - synthèse de haut-niveau
  - utilisation du modèle polyédrique
    - description à plusieurs niveaux d'abstraction (ALPHA)
    - fournit des transformations préservant la sémantique (MMALPHA)
- Lande :
  - conception d'outils d'aide au développement et à la validation de logiciels
  - utilisation de méthodes formelles

Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

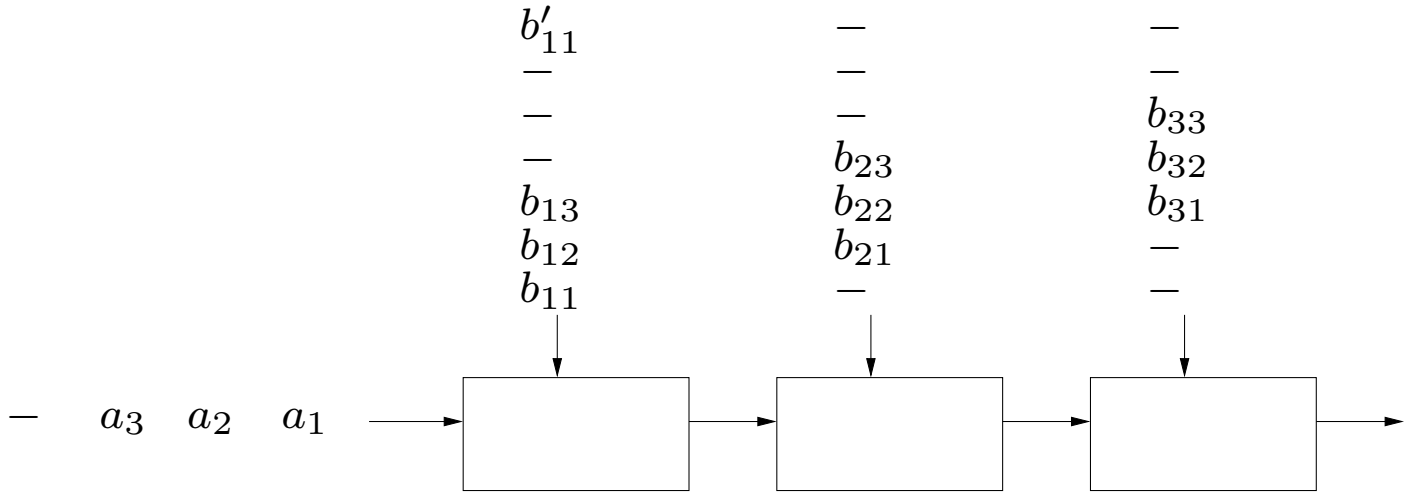
Exemple introductif

Exemple introductif

Exemple introductif

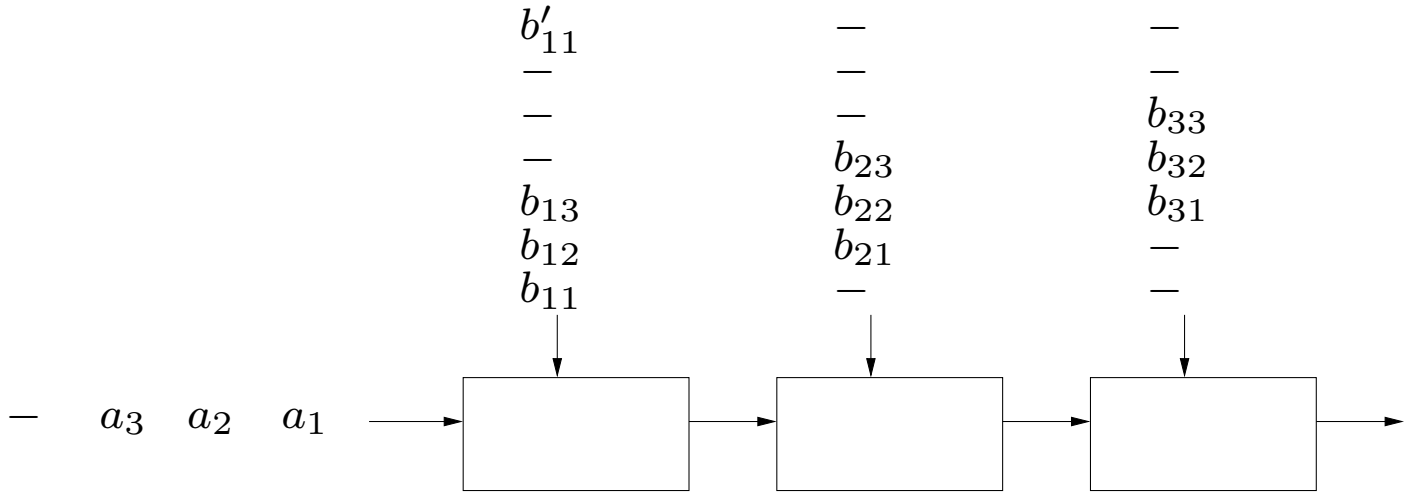
- Systèmes spécialisés, fonctionnent sans intervention humaine
- Traitement du signal, utilisation de filtres
- Exemple : calcul en chaîne de produits de matrices et de vecteurs

Produits de matrices vecteurs



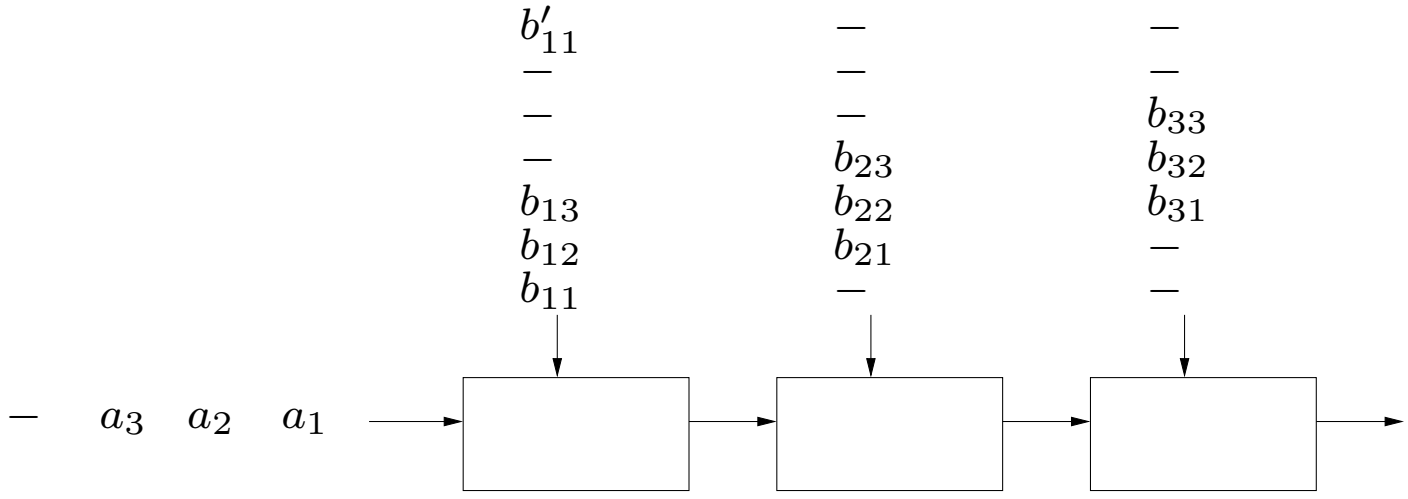
- Un réseau linéaire de N cellules

Produits de matrices vecteurs



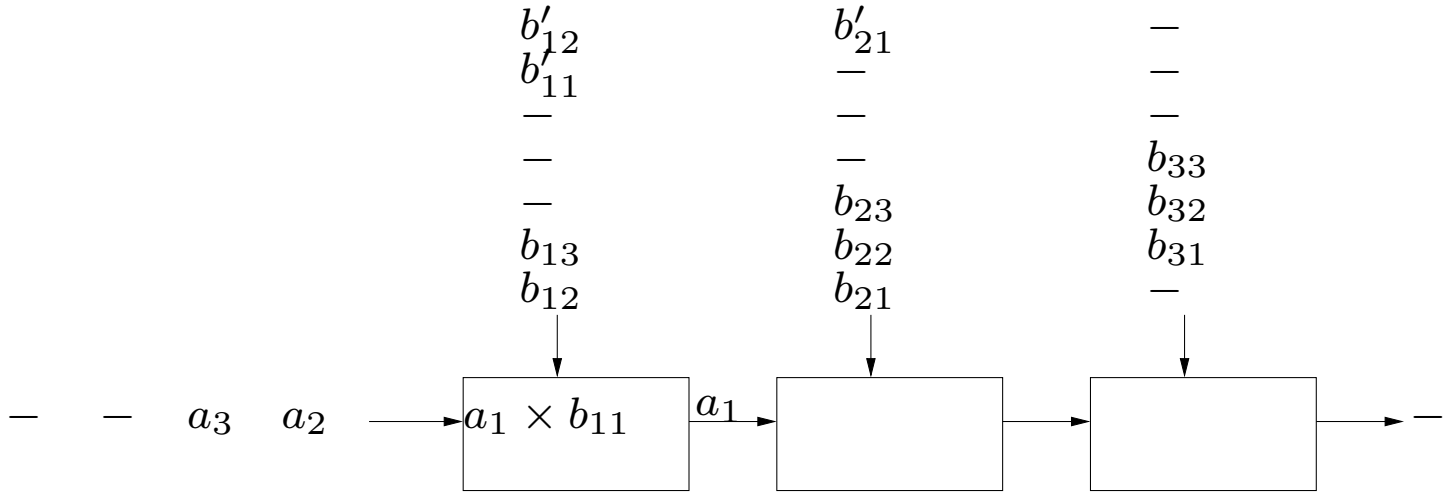
- Un réseau linéaire de N cellules
- Entrées a et b : suite de
  N coefficients suivis par N valeurs non significatives

Fonctionnement du système



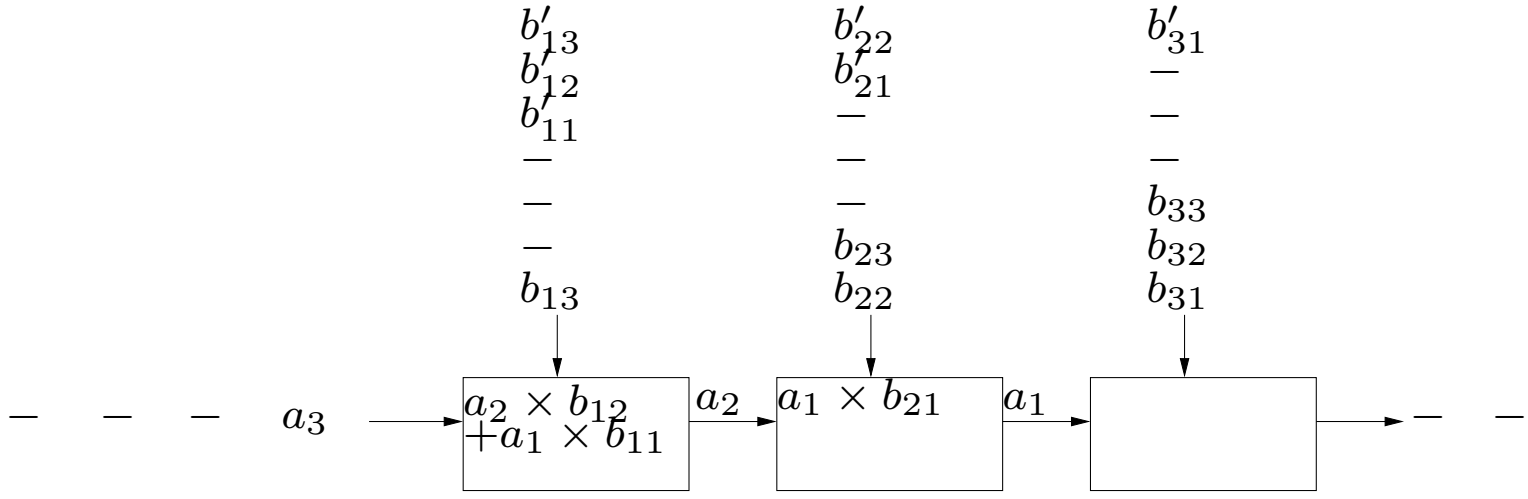
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système



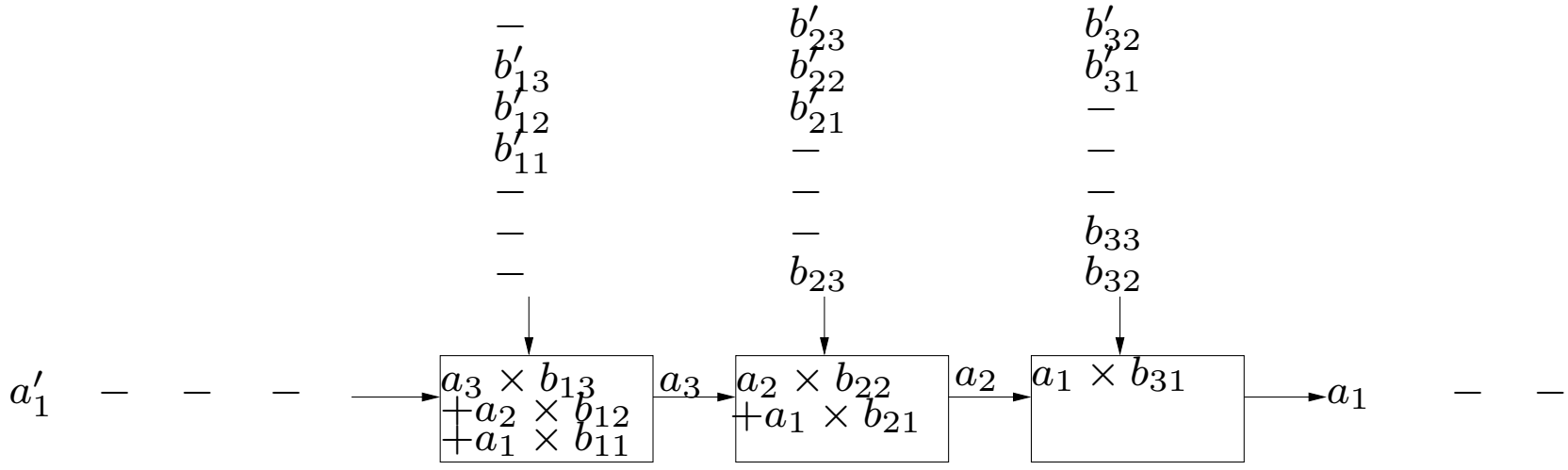
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système



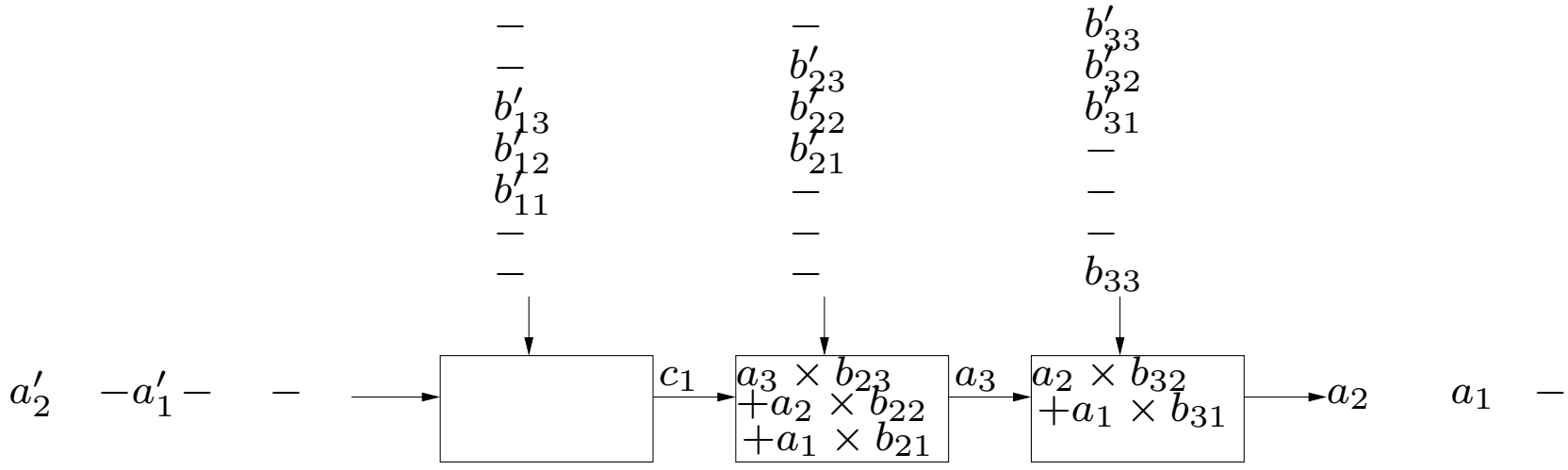
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système



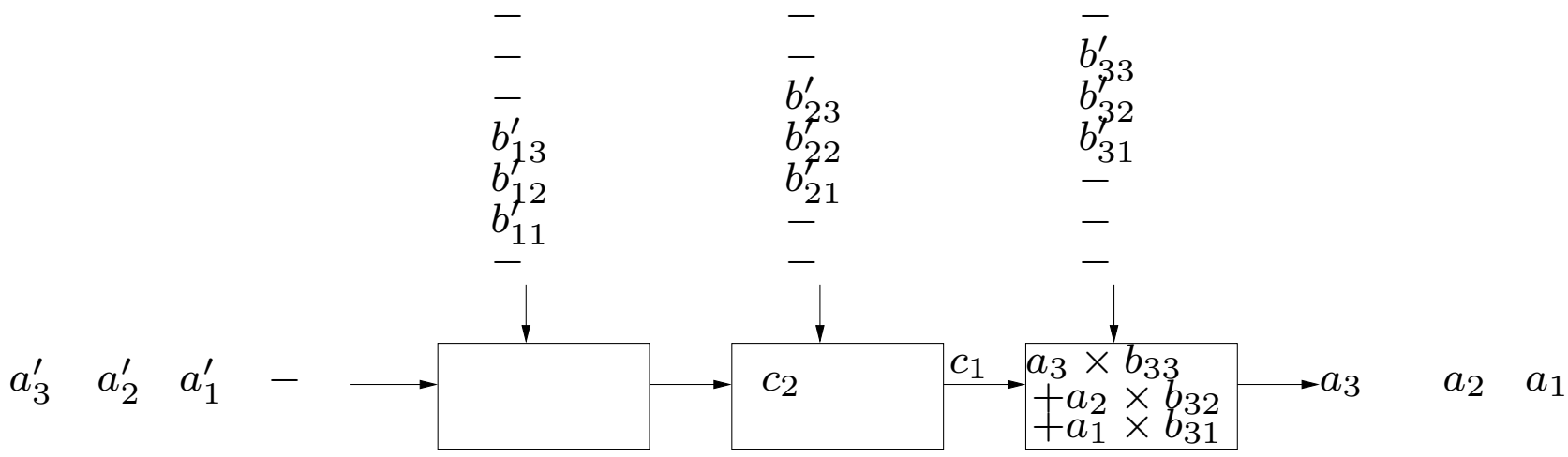
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système



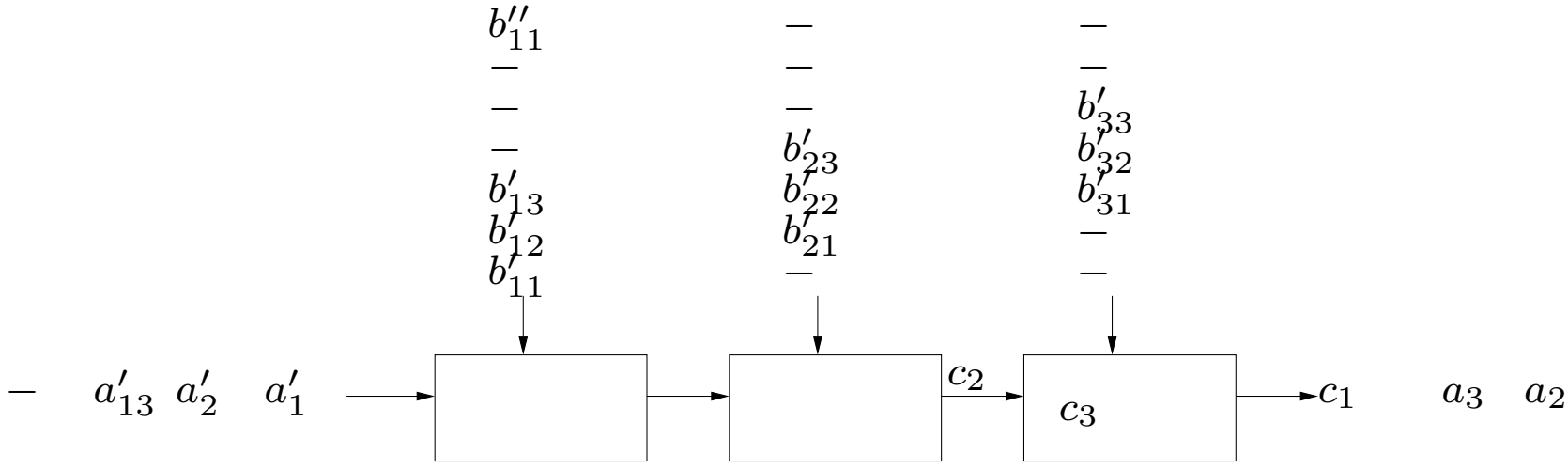
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système



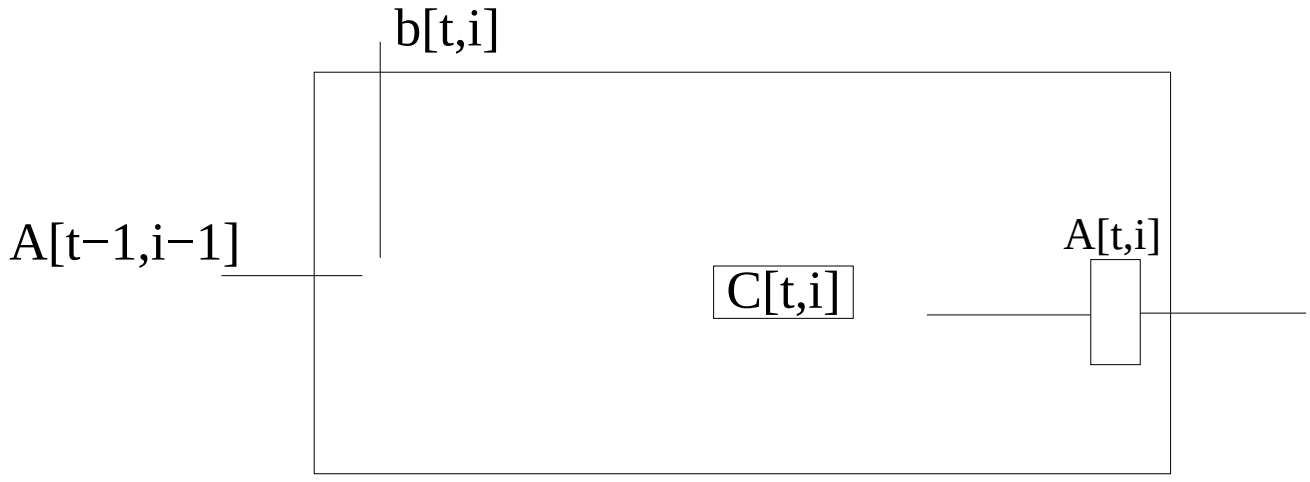
- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Fonctionnement du système

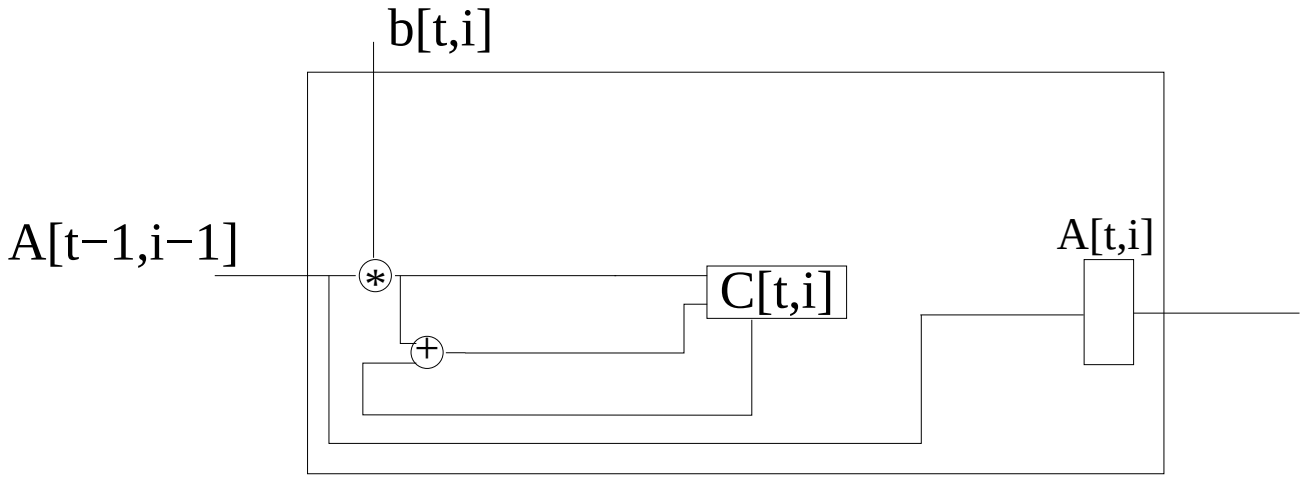


- Une cellule = un calcul d'un coefficient c de sortie
- À chaque instant, produit d'un coef. a par un coef. b qui est accumulé aux produits précédents
- Coef. a transmis de cellule en cellule
- Dès que coef. c calculé, la cellule est vidée et c transmis vers la sortie

Cellule i à l'instant t



Cellule i à l'instant t

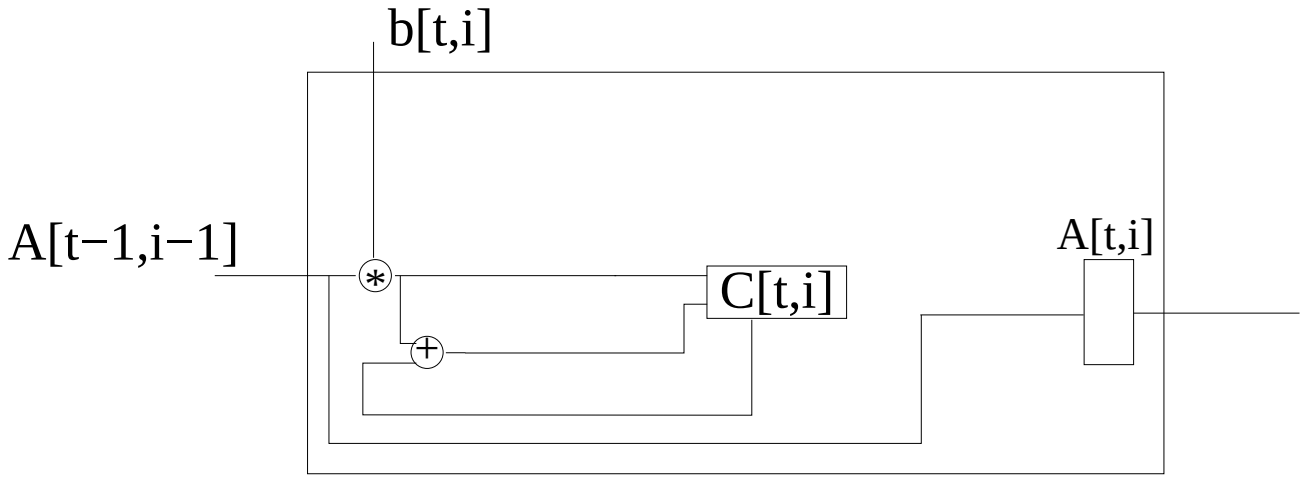


Équation récurrente définissant le registre C :

$$C[t,i] = C[t-1,i] + b[t,i] \times A[t-1,i-1]$$

- t est l'indice temporel
- i représente l'indice de la cellule

Cellule i à l'instant t



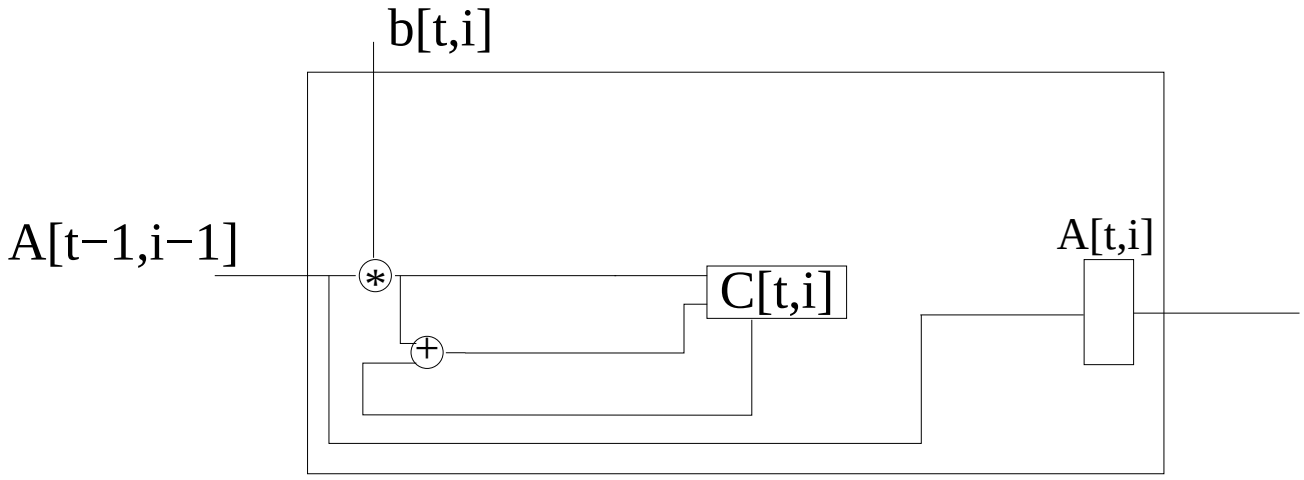
Équation récurrente définissant le registre C :

$$C[t,i] = C[t-1,i] + b[t,i] \times A[t-1,i-1]$$

Mais,

- b n'est pas toujours significatif et
- A :
  - soit coefficient du vecteur d'entrée,
  - soit coefficient du vecteur de sortie

Cellule i à l'instant t



Équation récurrente définissant le registre C :

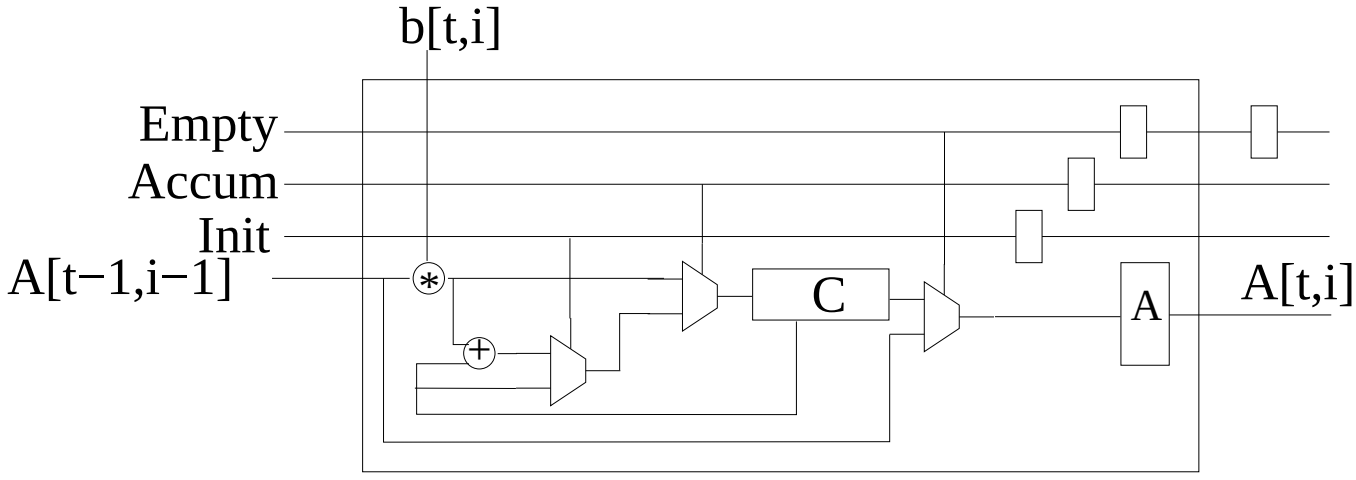
$$C[t,i] = C[t-1,i] + b[t,i] \times A[t-1,i-1]$$

Mais,

- b n'est pas toujours significatif et
- A :
  - soit coefficient du vecteur d'entrée,
  - soit coefficient du vecteur de sortie

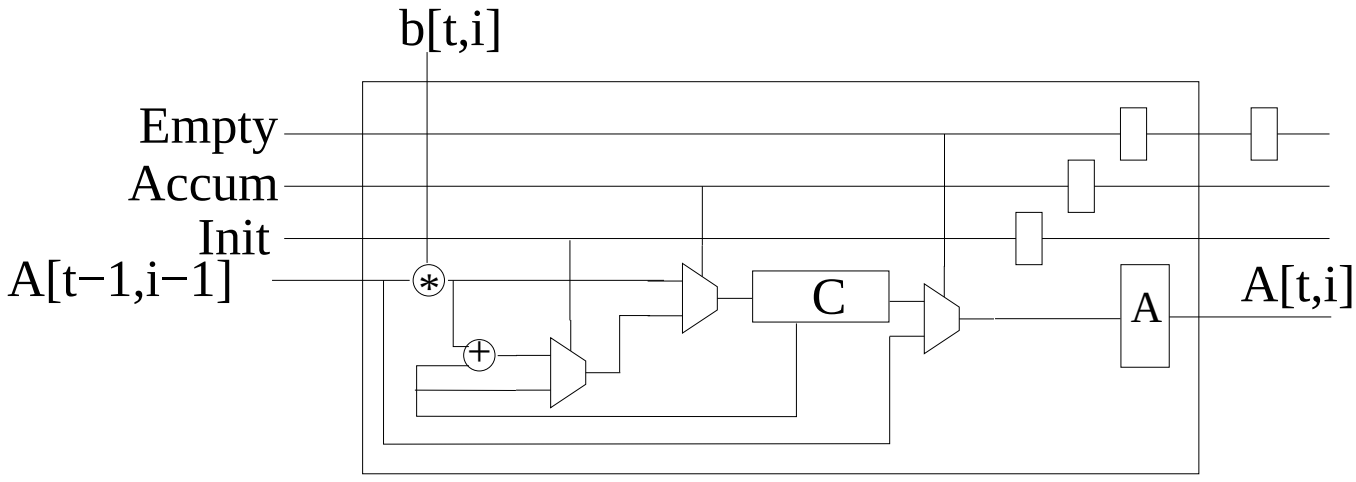
$\rightsquigarrow$ utilisation de signaux de contrôle

Cellule avec signaux de contrôle



- Init : initialise le registre C
- Accum : accumule les produits des coef.
- Empty : vide registre C dans registre A

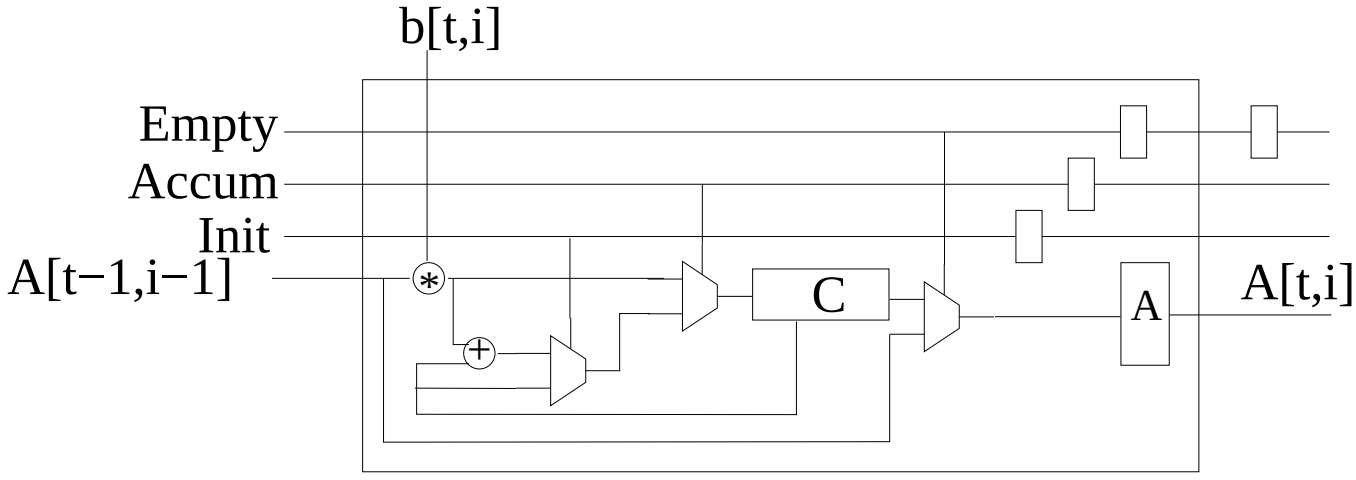
Cellule avec signaux de contrôle



Équation récurrente définissant le registre A :

$$\begin{aligned} A[t,i] = \quad & \text{if } \mathrm{empty}[t,i] \\ & \text{then } C[t-1,i] \\ & \text{else } A[t-1,i-1] \end{aligned}$$

Cellule avec signaux de contrôle

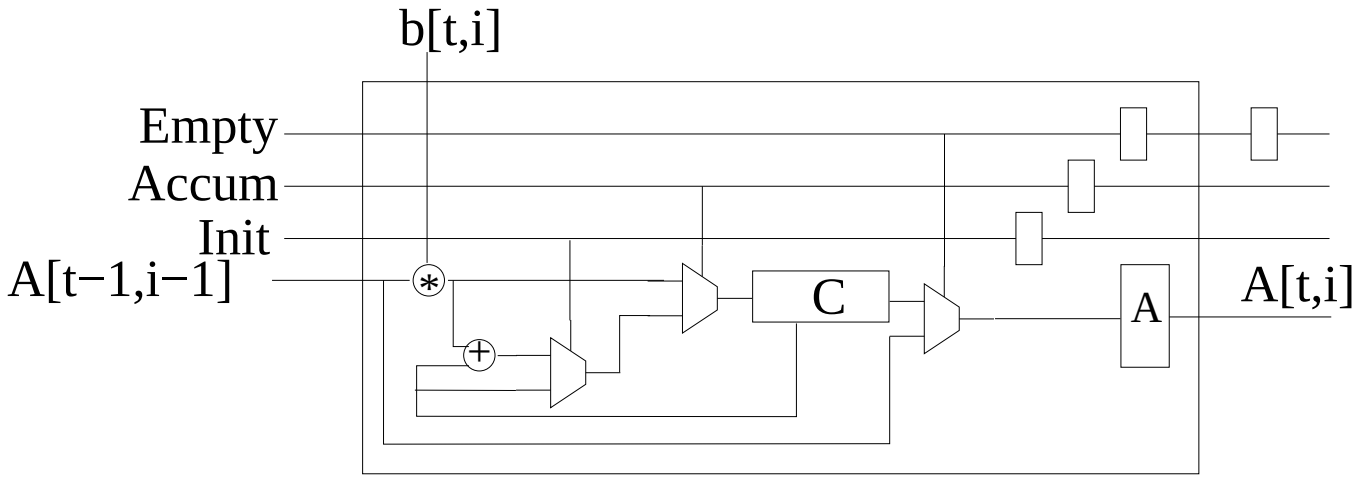


$$A[t,i] = \text{if empty}[t,i] \text{ then } C[t-1,i] \text{ else } A[t-1,i-1]$$

- Valeur de A pour $t = 0$?

$$A[0,i] = 0$$

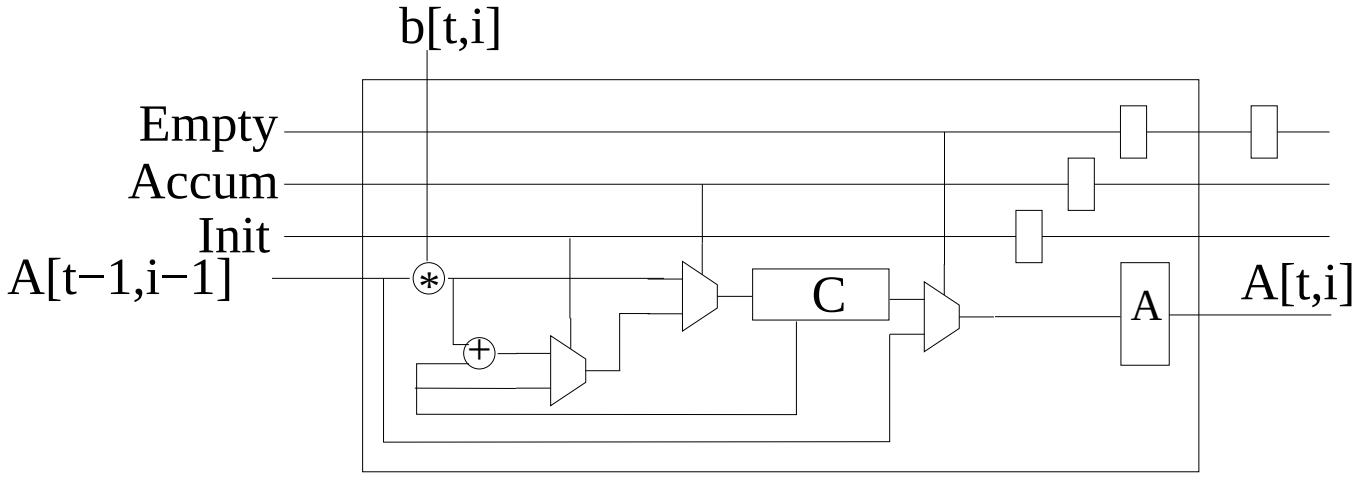
Cellule avec signaux de contrôle



$$A[t,i] = \text{if empty}[t,i] \text{ then } C[t-1,i] \text{ else } A[t-1,i-1]$$

- Valeur de A pour $t = 0$?
  $$A[0,i] = 0$$
- Valeur de A dans la première cellule, $t > 0$?
  $$A[t,0] = a[t]$$

Cellule avec signaux de contrôle



$$A[t,i] = \begin{cases} \{t,i \mid i = 0; t > 0\} : & a[t] \\ \{t,i \mid t = 0\} : & 0 \\ \{t,i \mid 1 \leq i; 1 \leq t\} : & \text{if empty}[t,i] \text{ then } C[t-1,i] \\ & \text{else } A[t-1,i-1] \end{cases}$$

Propriétés de sûreté

Ajout des signaux de contrôle difficile :

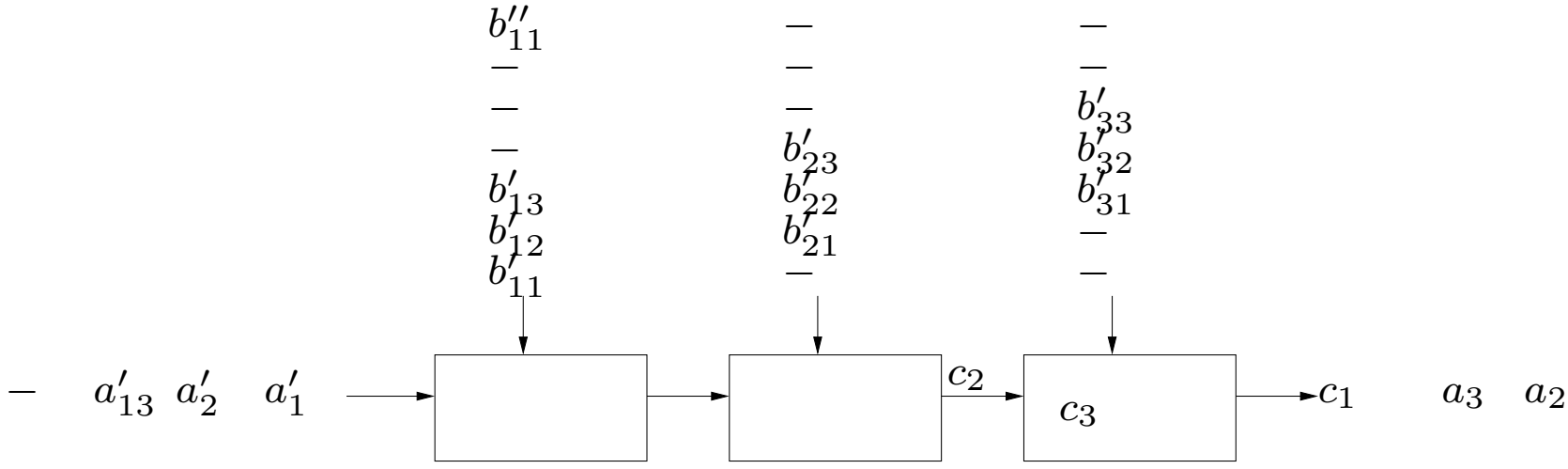
- soit ajoutés à la main
- soit par des transformations ne préservant pas la sémantique

$\rightsquigarrow$ nécessite vérification formelle

Ici, vérification de propriétés de sûreté

« Quelque chose de mauvais ne se produira jamais »

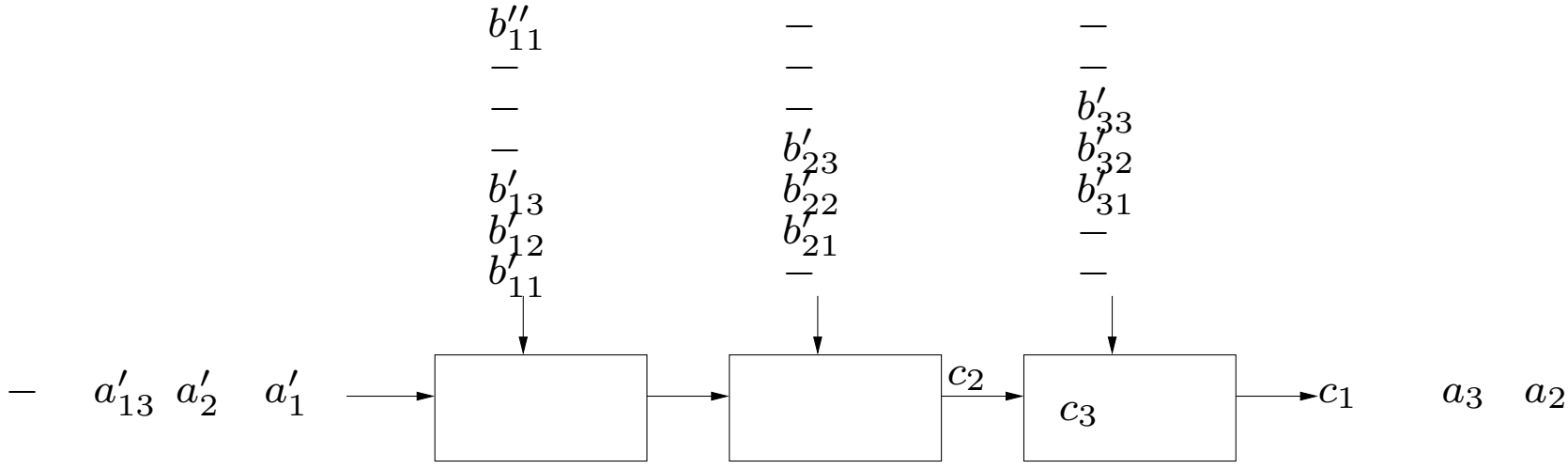
Exemple de propriétés



Résultat toujours significatif

- soit un coefficient du vecteur d'entrée
- soit un coefficient du vecteur de sortie

Exemple de propriétés

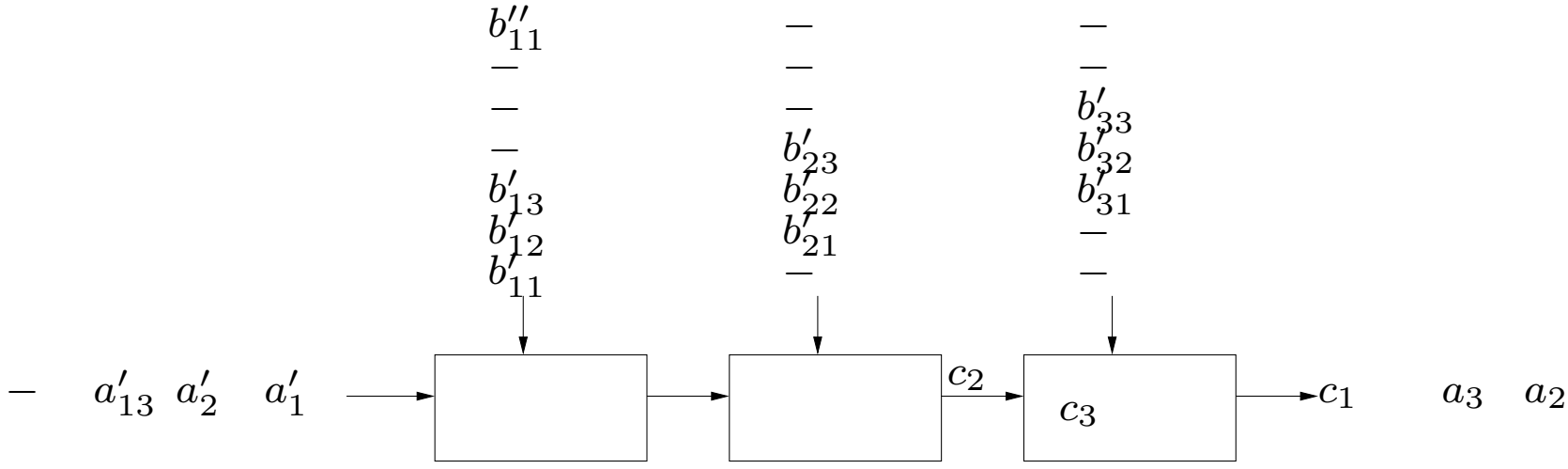


Résultat toujours significatif

- soit un coefficient du vecteur d'entrée
- soit un coefficient du vecteur de sortie

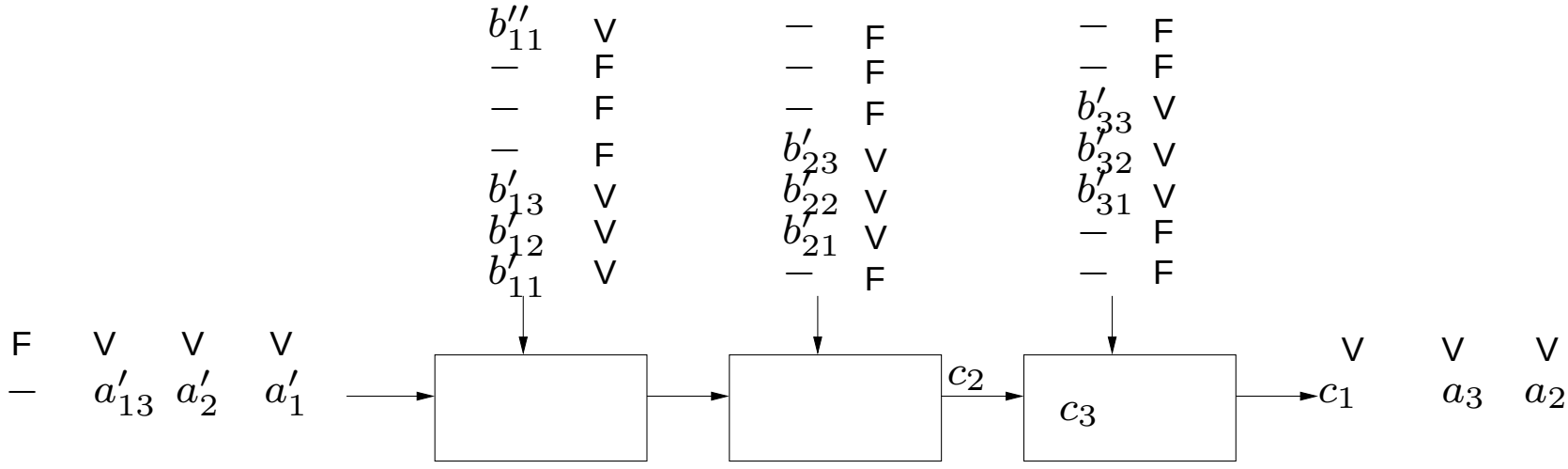
Restriction aux propriétés sur les signaux booléens

Abstraction du système



Propriété : les valeurs non significatives n'interviennent pas dans le résultat final.

Abstraction du système

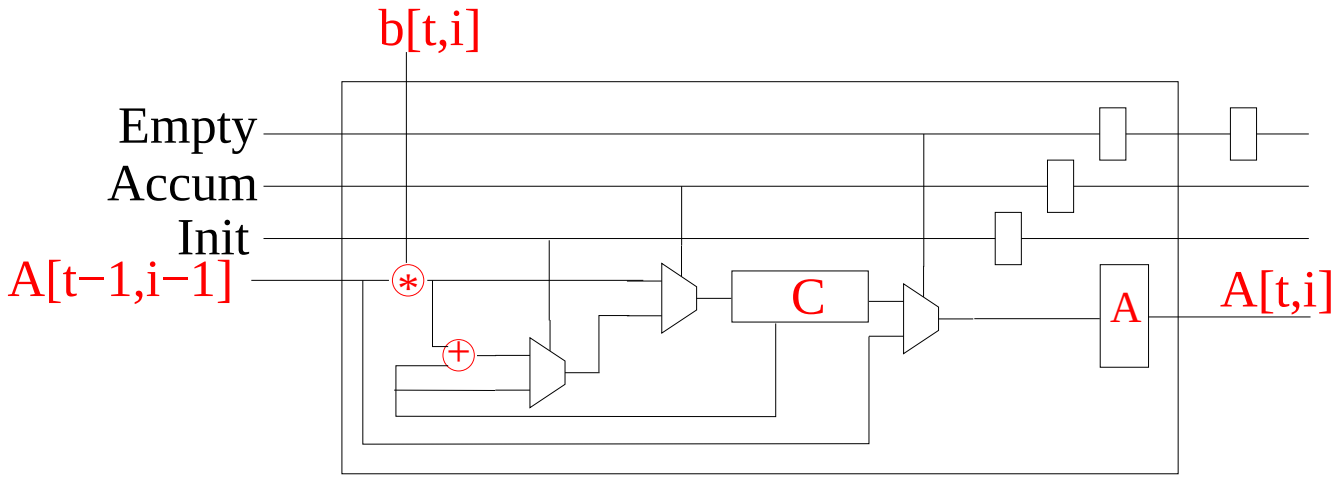


Propriété : les valeurs non significatives n'interviennent pas dans le résultat final.

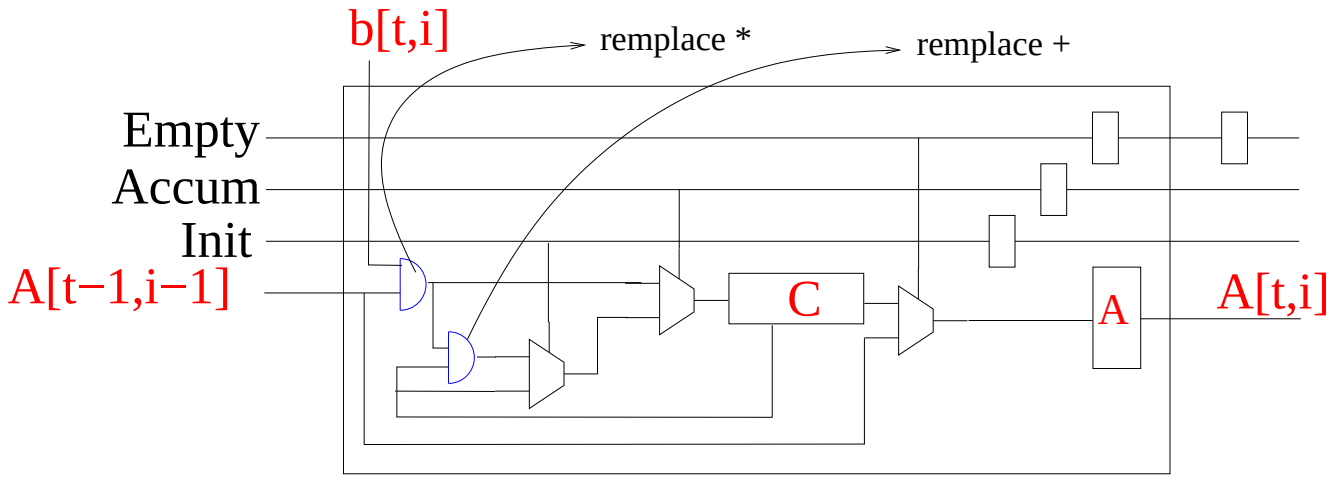
- Coefficients représentés par la valeur vrai
- Valeurs non significatives par la valeur faux

Propriété $\rightsquigarrow$ prouver que c vaut toujours vrai.

Cellule abstraite



Cellule abstraite



- Abstraction partageant la même structure globale (étape préliminaire)

Modèle polyédrique

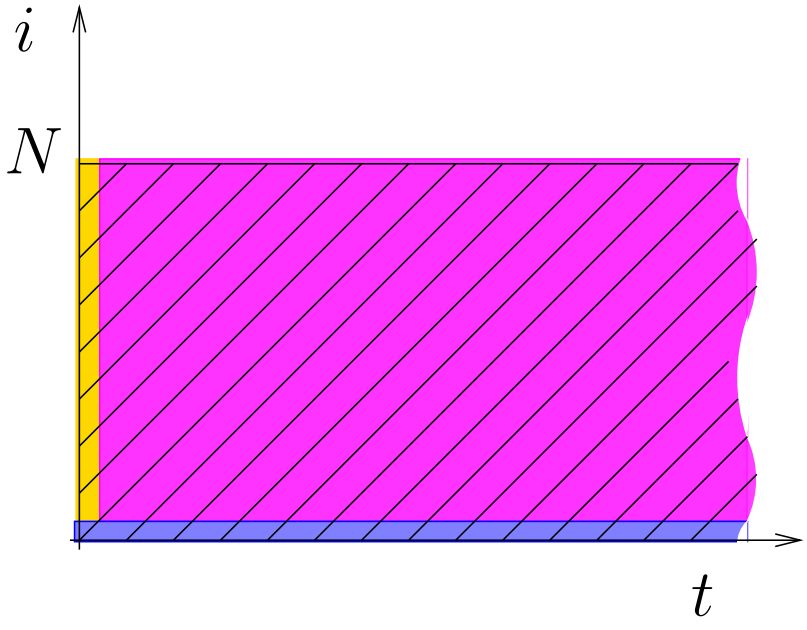
$$A[t,i] = \begin{cases} \{t,i \mid i = 0\} : & a[t] \\ \{t,i \mid t = 0\} : & \textit{False} \\ \{t,i \mid 1 \leq i; 1 \leq t\} : & \text{if empty[t, i] then C[t} - 1, \text{i]} \\ & \text{else A[t} - 1, \text{i} - 1] \end{cases}$$

équations récurrentes $+$

Modèle polyédrique

$$A[t,i] = \begin{cases} \{t,i \mid i = 0\} : & a[t] \\ \{t,i \mid t = 0\} : & \textit{False} \\ \{t,i \mid 1 \leq i; 1 \leq t\} : & \text{if empty}[t,i] \text{ then } C[t-1,i] \\ & \text{else } A[t-1,i-1] \end{cases}$$

équations récurrentes $+$domaines polyédriques

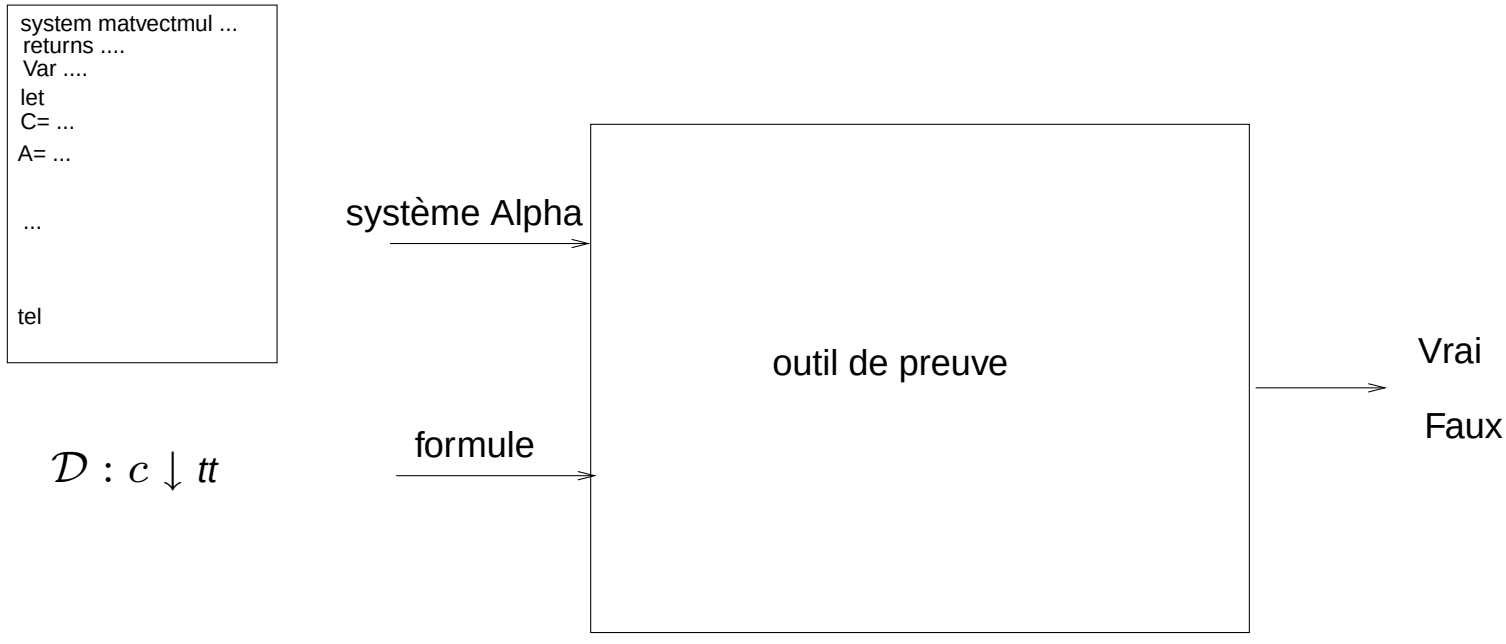


$a[t]$

0

If empty$[t,i]$ then $C[t-1,i]$
else $A[t-1,i-1]$

Objectif



Outil automatique de preuve de propriétés de sûreté sur des signaux de contrôle pour des systèmes décrits dans le modèle polyédrique

Travaux connexes

- Équivalence de 2 systèmes d'équations récurrentes
  - Travaux de Barthou, Feautrier, Redon : *On the equivalence of two systems of affine recurrence equations*
  - Travaux de Shashidar, Bruynooghe, Cathoor, Janssens : *Geometric Model Checking: An automatic verification technique for loop and data reuse transformations*
- Preuve de propriétés fonctionnelles
  - Travaux menés à l'IRISA par Cachera, Pichardie, Quinton, Rajopadhye, Risset

$\rightsquigarrow$ pas de travaux sur les propriétés de sûreté

Contributions

- Définition d'une logique polyédrique
- Algorithme de preuve (implémenté en MMALPHA)
- Utilisation de techniques d'agrandissement
- Publications :
  - Rapport de recherche : D.Cachera and K.Morin-Allory, *Verification of Control Properties in the Polyhedral Model*, Research Report 4756, INRIA, 2003.
  - Conférence : D.Cachera and K.Morin-Allory, *Verification of Control Properties in the Polyhedral Model*, In *Proceedings of Memocode 2003*, IEEE, Mont-St-Michel, France, juin 2003.
  - Article : D.Cachera and K.Morin-Allory, *Verification of Safety Properties for Parameterized Systems*, *Transactions on Embedded Computing Systems*, ACM, sous presse

Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

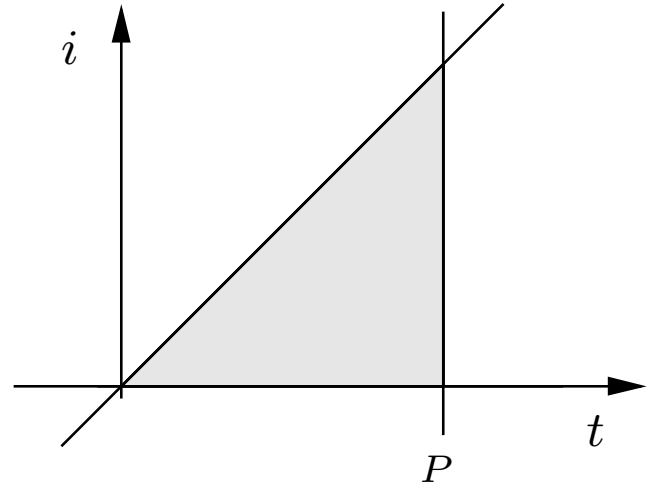
Polyèdres

Polyèdre : sous ensemble de $\mathbb{Q}^n$ (rationnels) délimité par un nombre fini d'hyper-plans.

Polyèdres

Polyèdre : sous ensemble de $\mathbb{Q}^n$ (rationnels) délimité par un nombre fini d'hyper-plans.

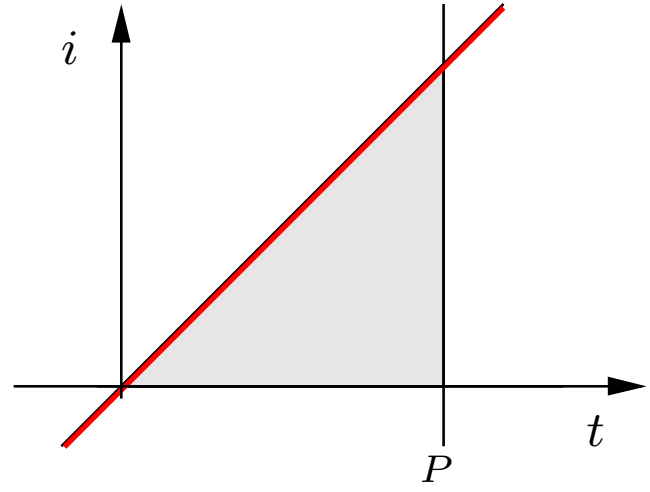
$$\mathcal{P} = \{t, i \in \mathbb{Q} \mid 0 \leq i \leq t \leq P\}$$



Polyèdres

Polyèdre : sous ensemble de $\mathbb{Q}^n$ (rationnels) délimité par un nombre fini d'hyper-plans.

$$\mathcal{P} = \{t, i \in \mathbb{Q} \mid 0 \leq i \leq t \leq P\}$$

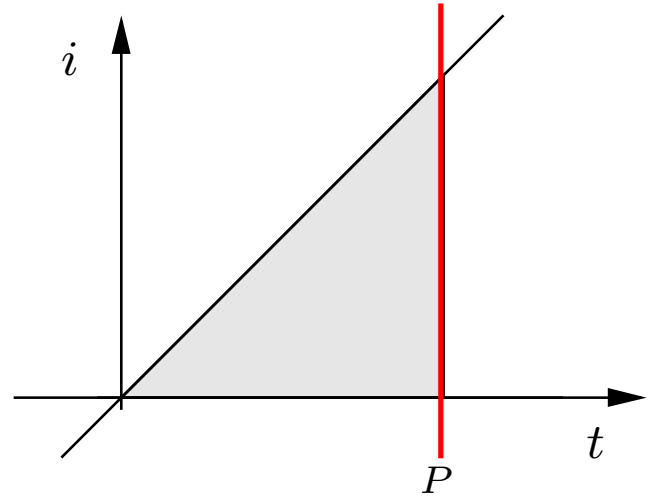


- la droite $\{t, i \mid t - i = 0\}$,

Polyèdres

Polyèdre : sous ensemble de $\mathbb{Q}^n$ (rationnels) délimité par un nombre fini d'hyper-plans.

$$\mathcal{P} = \{t, i \in \mathbb{Q} \mid 0 \leq i \leq t \leq P\}$$

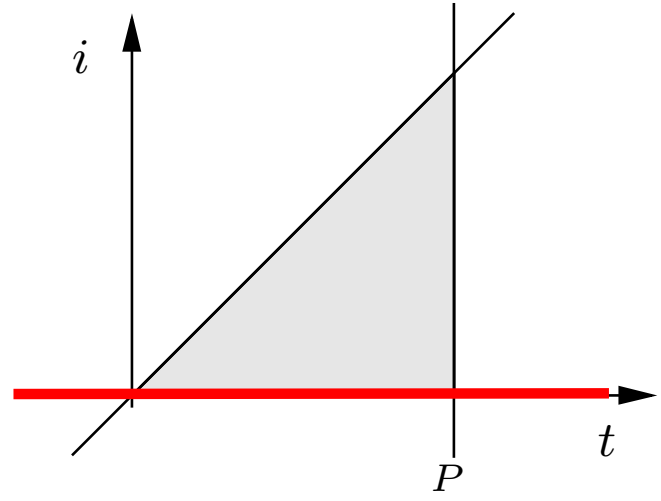


- la droite $\{t, i \mid t - i = 0\}$,
- la droite $\{t, i \mid t - P = 0\}$ et

Polyèdres

Polyèdre : sous ensemble de $\mathbb{Q}^n$ (rationnels) délimité par un nombre fini d'hyper-plans.

$$\mathcal{P} = \{t, i \in \mathbb{Q} \mid 0 \leq i \leq t \leq P\}$$



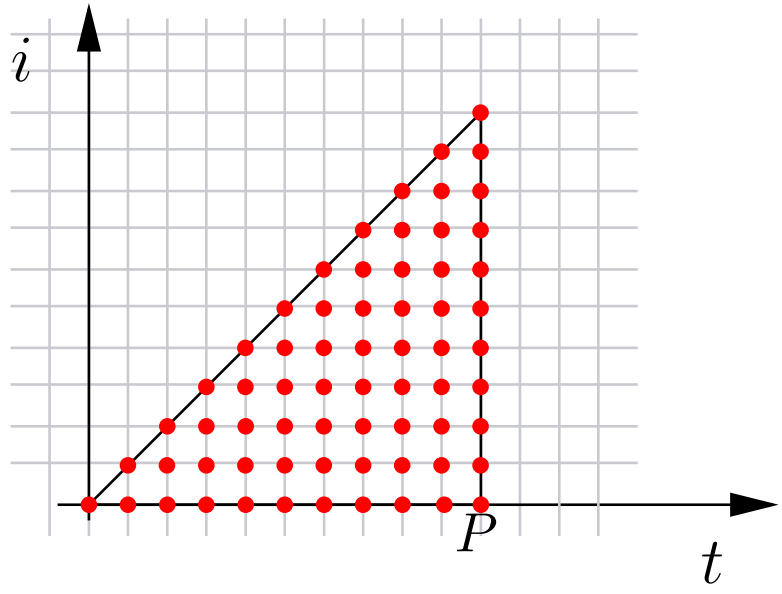
- la droite $\{t, i \mid t - i = 0\}$,
- la droite $\{t, i \mid t - P = 0\}$ et
- la droite $\{t, i \mid i = 0\}$.

Polyèdres à points entiers

Polyèdre à points entiers : intersection d'un polyèdre avec $\mathbb{Z}^n$.

Polyèdres à points entiers

Polyèdre à points entiers : intersection d'un polyèdre avec $\mathbb{Z}^n$.



$$\mathcal{P} = \{t, i \in \mathbb{Z} \mid 0 \leq i \leq t \leq P\}$$

Variable Polyédrique

- Variable polyédrique X : application

$$X :$$

Variable Polyédrique

- Variable polyédrique X : application
  - d'un domaine $\mathcal{D}$: union finie de polyèdres à points entiers

$$X : \mathcal{D}$$

Variable Polyédrique

- Variable polyédrique X : application
  - d'un domaine $\mathcal{D}$: union finie de polyèdres à points entiers
  - dans un ensemble de valeurs (ici, les booléens $\mathbb{B}$)

$$X : \mathcal{D} \rightarrow \mathbb{B}$$

Variable Polyédrique

- Variable polyédrique X : application
  - d'un domaine $\mathcal{D}$: union finie de polyèdres à points entiers
  - dans un ensemble de valeurs (ici, les booléens $\mathbb{B}$)

$$X : \mathcal{D} \rightarrow \mathbb{B}$$

- Instance : restriction d'une variable à un point de son domaine (exemple : valeur de $A[t, i]$ = valeur du registre A à l'instant t dans la cellule i)

Fonctions de dépendance

$$C[t,i] = C[t-1,i] \wedge A[t-1,i-1] \wedge b[t,i]$$

- Notation :
  $C = C.(t,i \longrightarrow t-1,i) \wedge A.(t,i \longrightarrow t-1,i-1) \wedge b.(t,i \longrightarrow t,i)$
  - dépendance affine : application linéaire
  - dépendance uniforme : translation
  - auto-dépendance : C dépend de C.
- Intérêt : composition de dépendances
  $C.d = (C.(t,i \longrightarrow t-1,i)).d \wedge (A.(t,i \longrightarrow t-1,i-1)).d \wedge (b.(t,i \longrightarrow t,i)).d$
- calculs symboliques sur les polyèdres et les dépendances : utilisation de la bibliothèque polyédrique PolyLib

Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

Logique Polyédrique

- Objectif : prouver qu'un signal vaut vrai (ou faux) à tout instant et pour tout processeur (exemple : prouver que le signal c après N instants vaut toujours vrai)
- Traduction dans le modèle polyédrique : prouver qu'une expression vaut vrai (ou faux) sur tout son domaine (exemple : prouver que la variable c vaut vrai sur le domaine $\{t \mid t \geq N\}$)

Utilisation de la sémantique dénotationnelle de ALPHA (langage fournissant la syntaxe des équations récurrentes affines, non développée ici).

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver
- Formule :

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver
- Formule :
  - un domaine $\mathcal{D}$,

$\mathcal{D}$:

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver
- Formule :
  - un domaine $\mathcal{D}$,
  - une expression polyédrique e et

$$\mathcal{D} : e$$

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver
- Formule :
  - un domaine $\mathcal{D}$,
  - une expression polyédrique e et
  - une valeur booléenne v.

$$\mathcal{D} : e \downarrow v$$

Syntaxe des formules

- Syntaxe induite par le type de propriétés à prouver
- Formule :
  - un domaine $\mathcal{D}$,
  - une expression polyédrique e et
  - une valeur booléenne v.

$$\mathcal{D} : e \downarrow v$$

- Restriction :
  - $(\mathcal{D} : e \downarrow v) \vee (\mathcal{D}' : e' \downarrow v')$ pas étudiée ici
  - mais, $\mathcal{D} : (e \vee e') \downarrow v$ sera étudiée

Système de preuve

- Pour un système S, prouver que e vaut v en tout point de $\mathcal{D}$ est indécidable (Apt et Kozen en 86) car $\mathcal{D}$ et S paramétrés $\rightsquigarrow$ construction d'un système de règles de preuve.

Système de preuve

- Pour un système S, prouver que e vaut v en tout point de $\mathcal{D}$ est indécidable (Apt et Kozen en 86) car $\mathcal{D}$ et S paramétrés $\rightsquigarrow$ construction d'un système de règles de preuve.
- Notation :

$$S \vdash \mathcal{D} : e \downarrow v$$

Système de preuve

- Pour un système S, prouver que e vaut v en tout point de $\mathcal{D}$ est indécidable (Apt et Kozen en 86) car $\mathcal{D}$ et S paramétrés $\rightsquigarrow$ construction d'un système de règles de preuve.
- Notation :

$$S \vdash \mathcal{D} : e \downarrow v$$

  - Se lit normalement : il existe une preuve de la formule $\mathcal{D} : e \downarrow v$ pour le système S
  - Mais sera lu : il existe une preuve que e vaut v sur le domaine $\mathcal{D}$ (dans la sémantique dénotationnelle)

Système de preuve

- Pour un système S, prouver que e vaut v en tout point de $\mathcal{D}$ est indécidable (Apt et Kozen en 86) car $\mathcal{D}$ et S paramétrés $\rightsquigarrow$ construction d'un système de règles de preuve.
- Notation :

$$S \vdash \mathcal{D} : e \downarrow v$$

  - Se lit normalement : il existe une preuve de la formule $\mathcal{D} : e \downarrow v$ pour le système S
  - Mais sera lu : il existe une preuve que e vaut v sur le domaine $\mathcal{D}$ (dans la sémantique dénotationnelle)
- S'il existe une preuve de $\mathcal{D} : e \downarrow v$ pour le système S alors en tout point de $\mathcal{D}$, la valeur de l'expression e dans la sémantique dénotationnelle est v.

Systèmes de règles de preuve

Différentes règles de preuve :

- Règles classiques sur les connecteurs logiques
- Règle de substitution par constante
- Règle de recherche de motifs
- Règle de recherche d'auto-dépendances

Un axiome

$$\frac{}{S \vdash \mathcal{D} : \mathit{True} \downarrow \mathit{tt}}\ \textsc{ax}1$$

Règle d'introduction du "et"

$$\frac{S \vdash \mathcal{D} : e_1 \downarrow \mathit{tt}, \quad S \vdash \mathcal{D} : e_2 \downarrow \mathit{tt}}{S \vdash \mathcal{D} : e_1 \wedge e_2 \downarrow \mathit{tt}}\ \text{ET}$$

À propos de la disjonction

$$\frac{S \vdash \mathcal{D} : e_1 \downarrow tt}{S \vdash \mathcal{D} : e_1 \vee e_2 \downarrow tt}\ \text{OU1} \qquad \frac{S \vdash \mathcal{D} : e_2 \downarrow tt}{S \vdash \mathcal{D} : e_1 \vee e_2 \downarrow tt}\ \text{OU2}$$

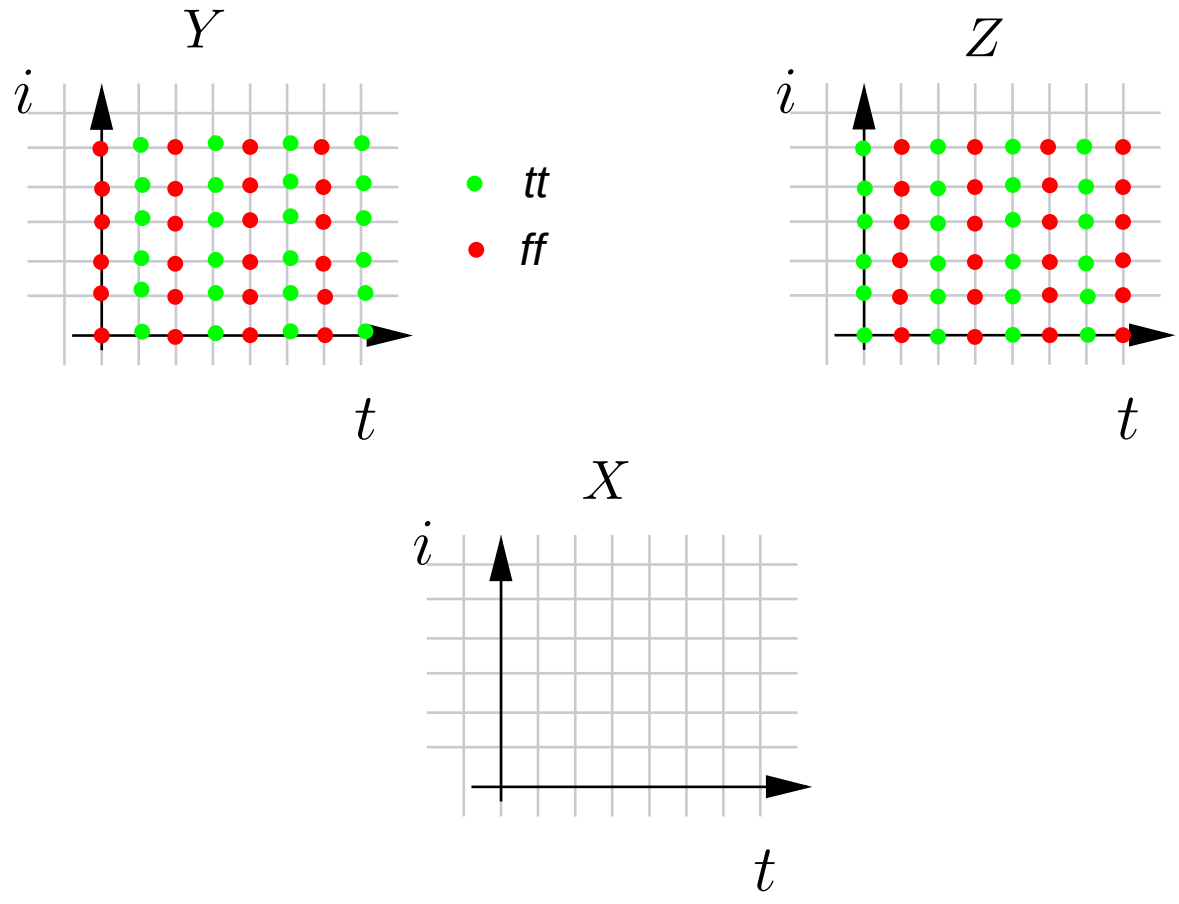
Règle sûre, mais trop restrictive

La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow \mathit{tt}$.

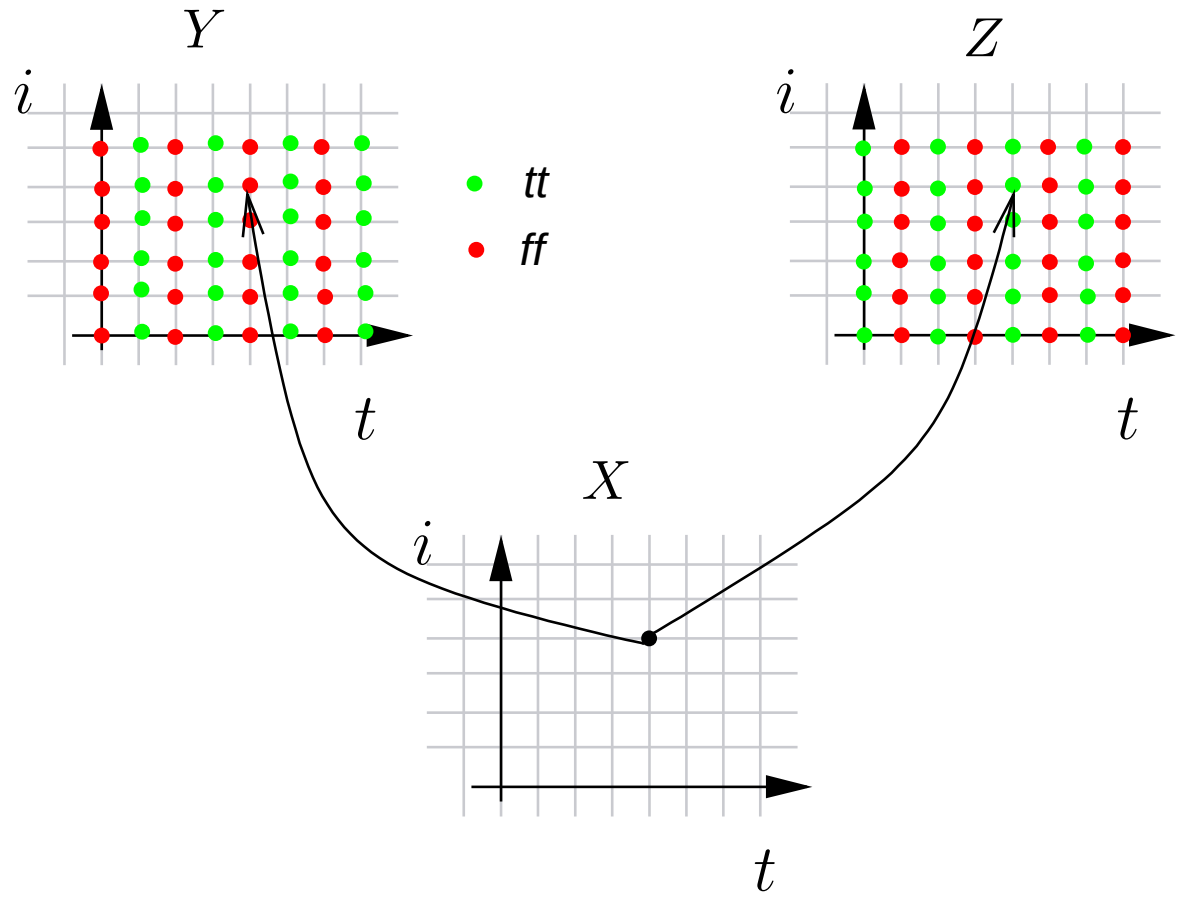
La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.



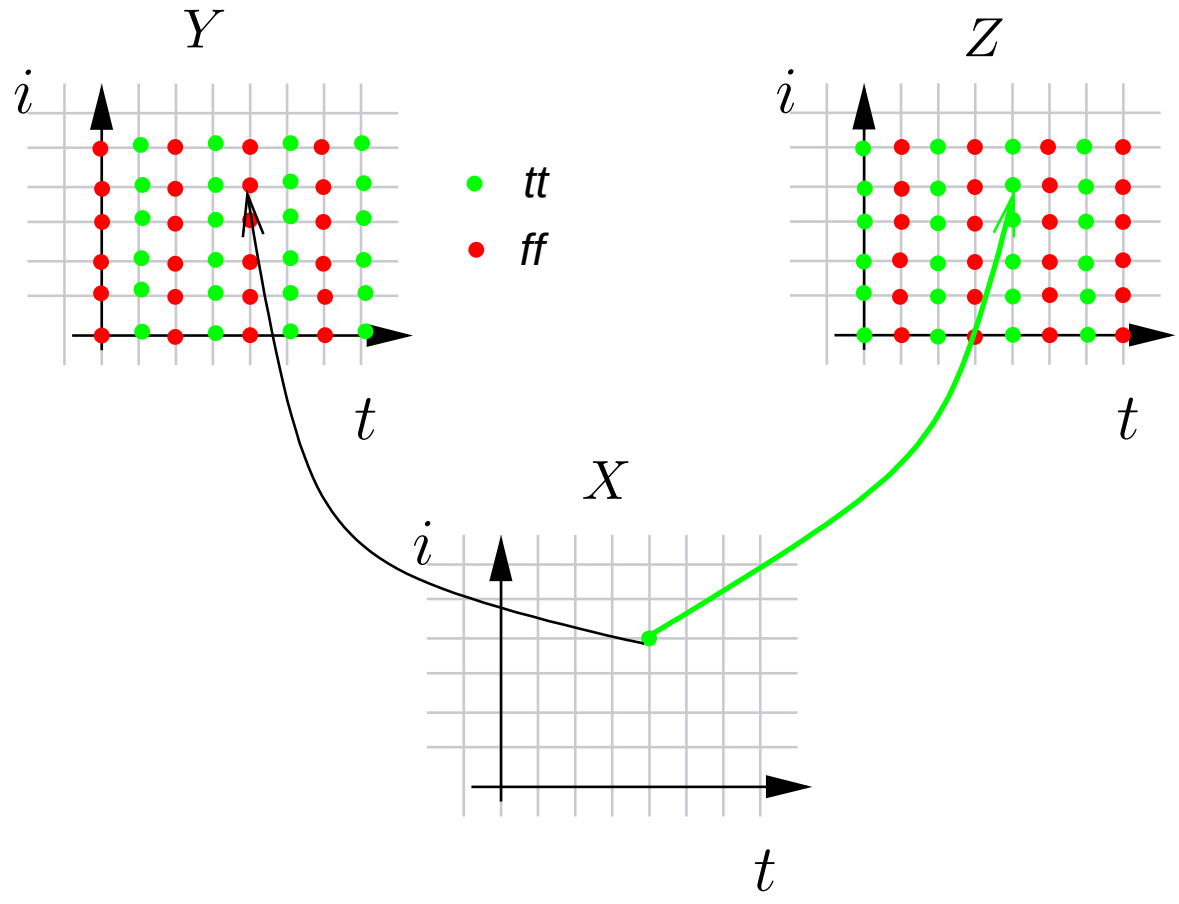
La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.



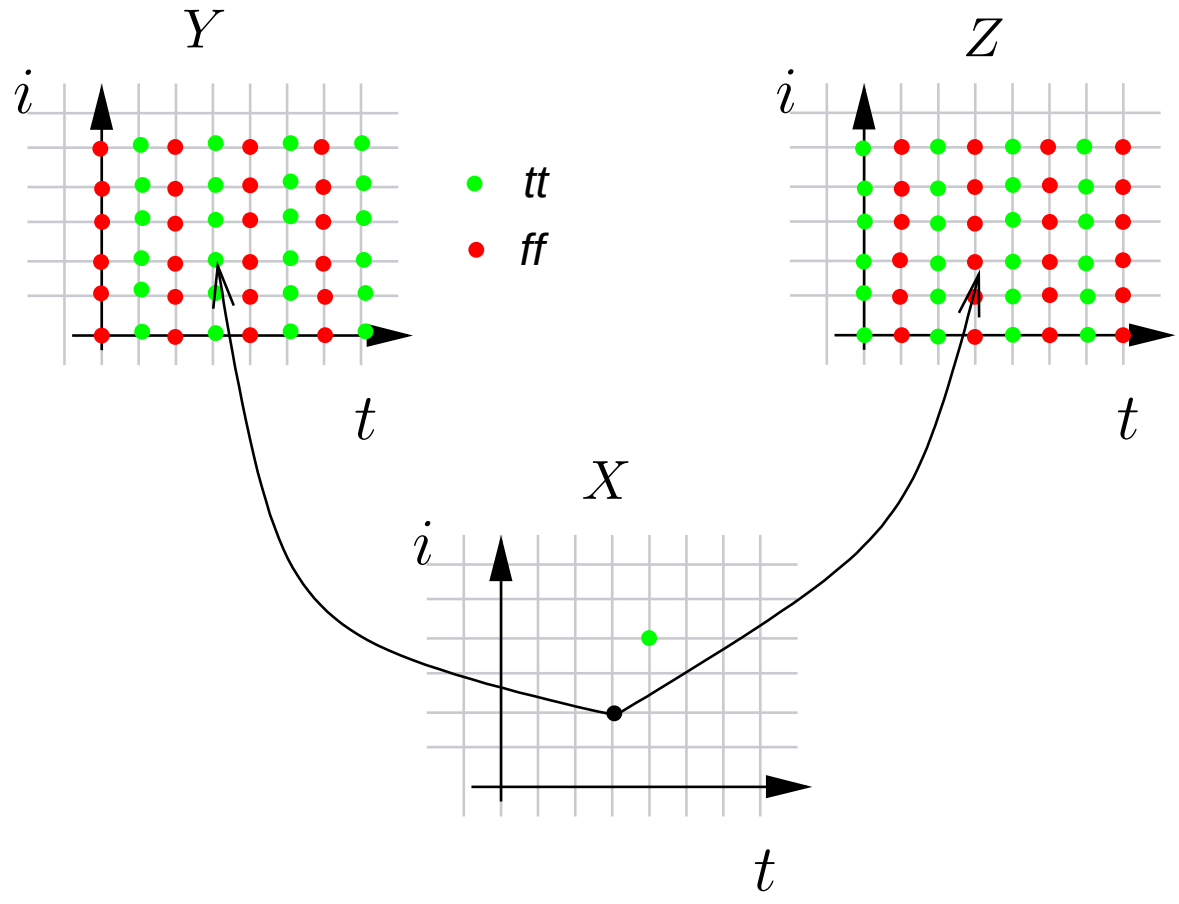
La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.



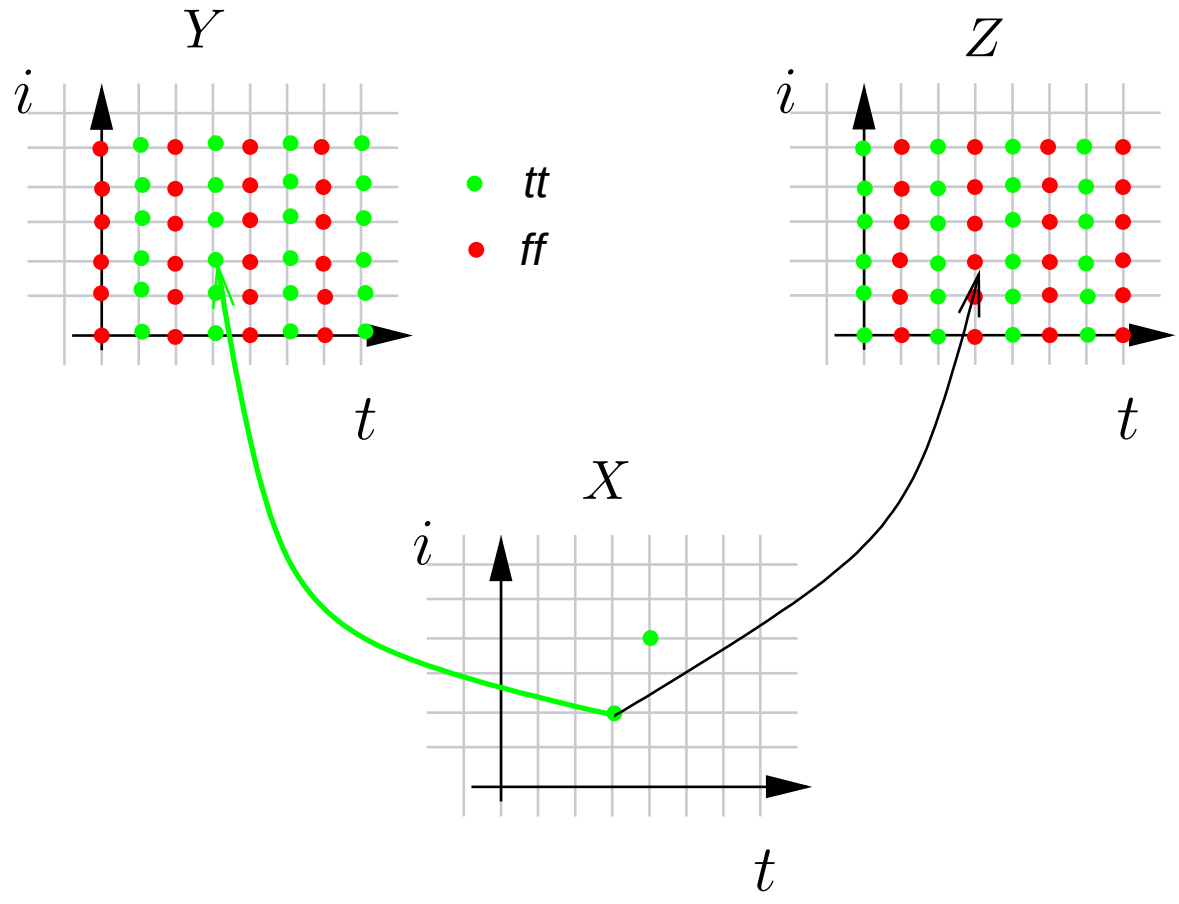
La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.



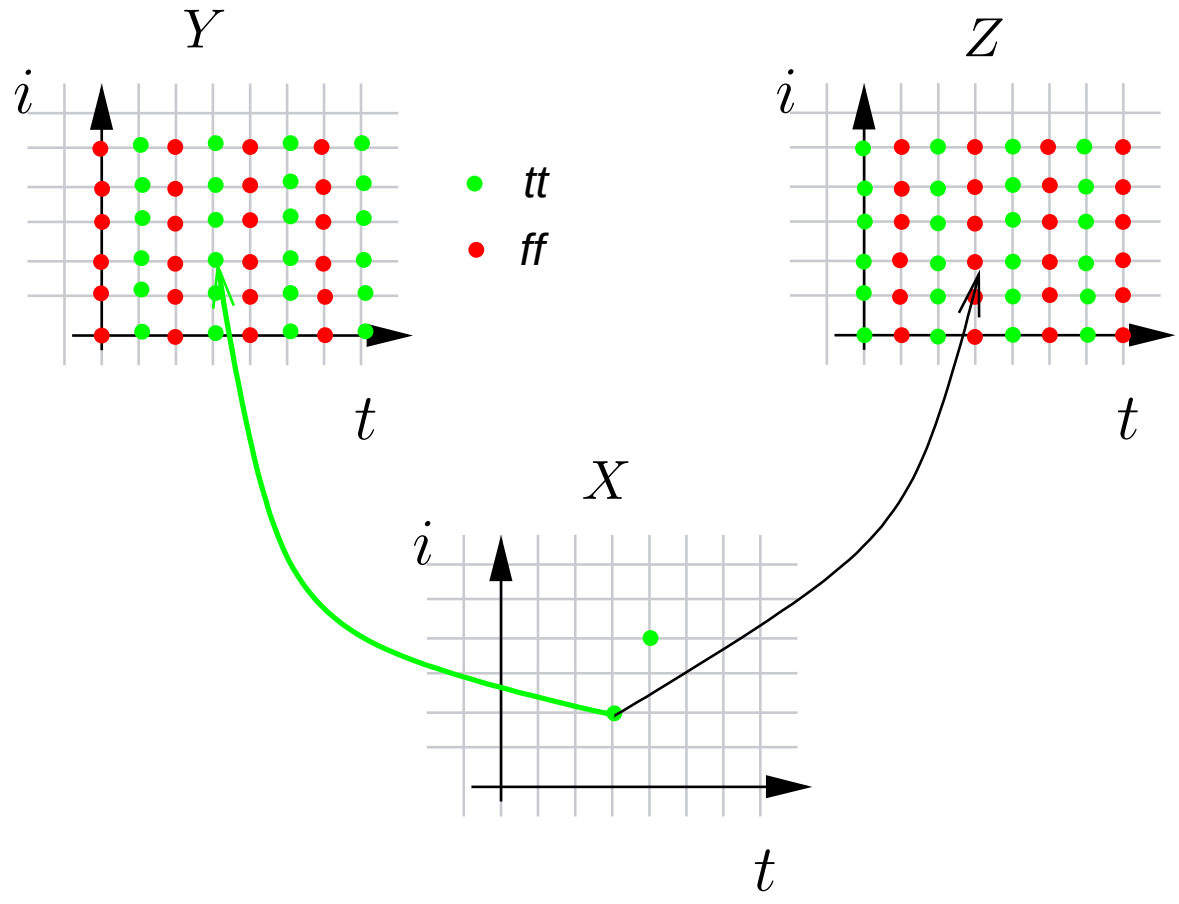
La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.



La disjonction

Soit $X = Y \vee Z$, on veut prouver $\mathcal{D} : X \downarrow tt$.

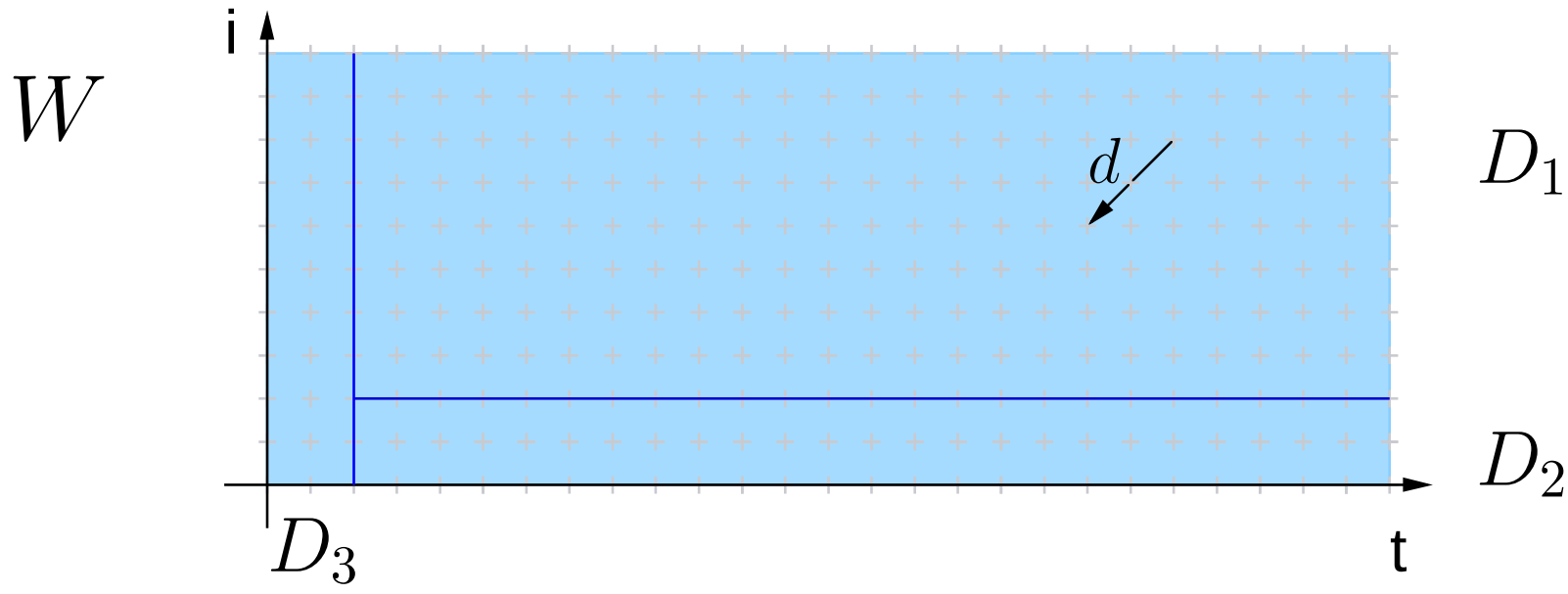


$\rightsquigarrow$ d'autres règles de preuve.

Substitution par constante (1)

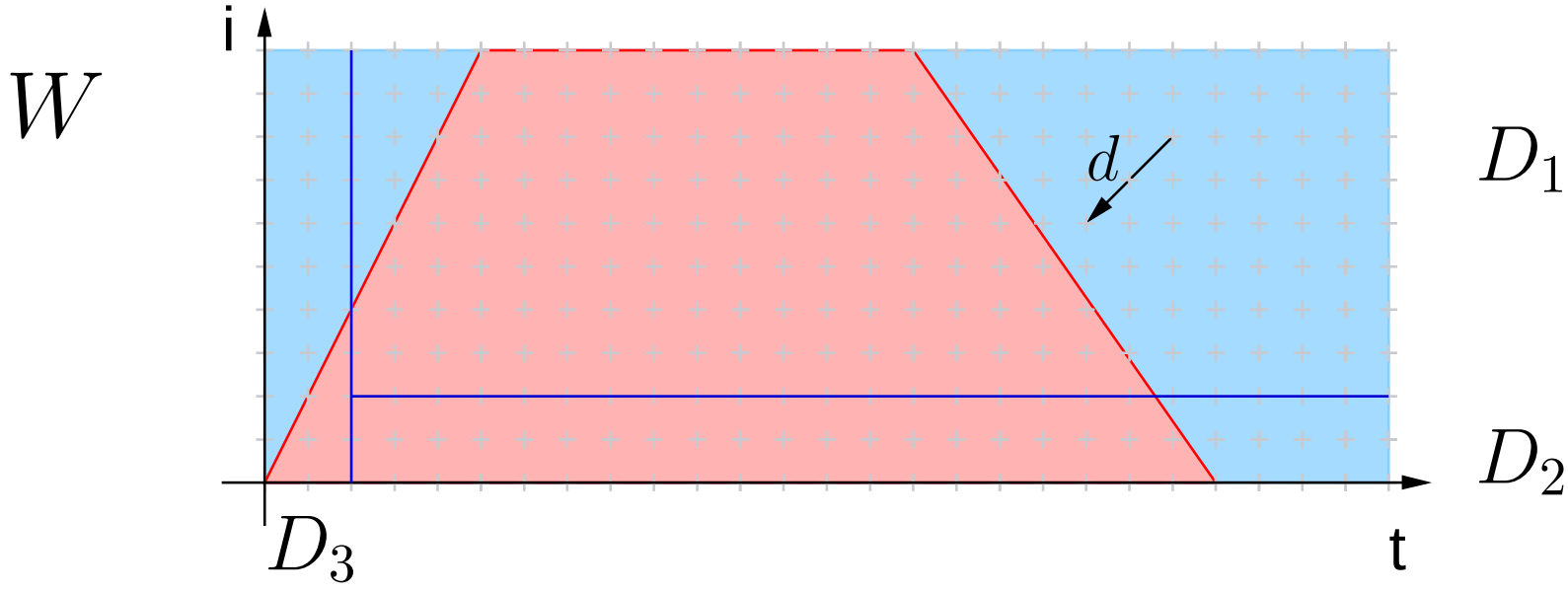
- But : prouver $\mathcal{D} : W \downarrow \mathit{tt}$
- W définie à l'aide d'auto-dépendance ($W = W.d \vee \ldots$)
- $\rightsquigarrow$ simplifier l'écriture de W
- Technique : substituer les auto-dépendances par *True*

Substitution par constante (2)



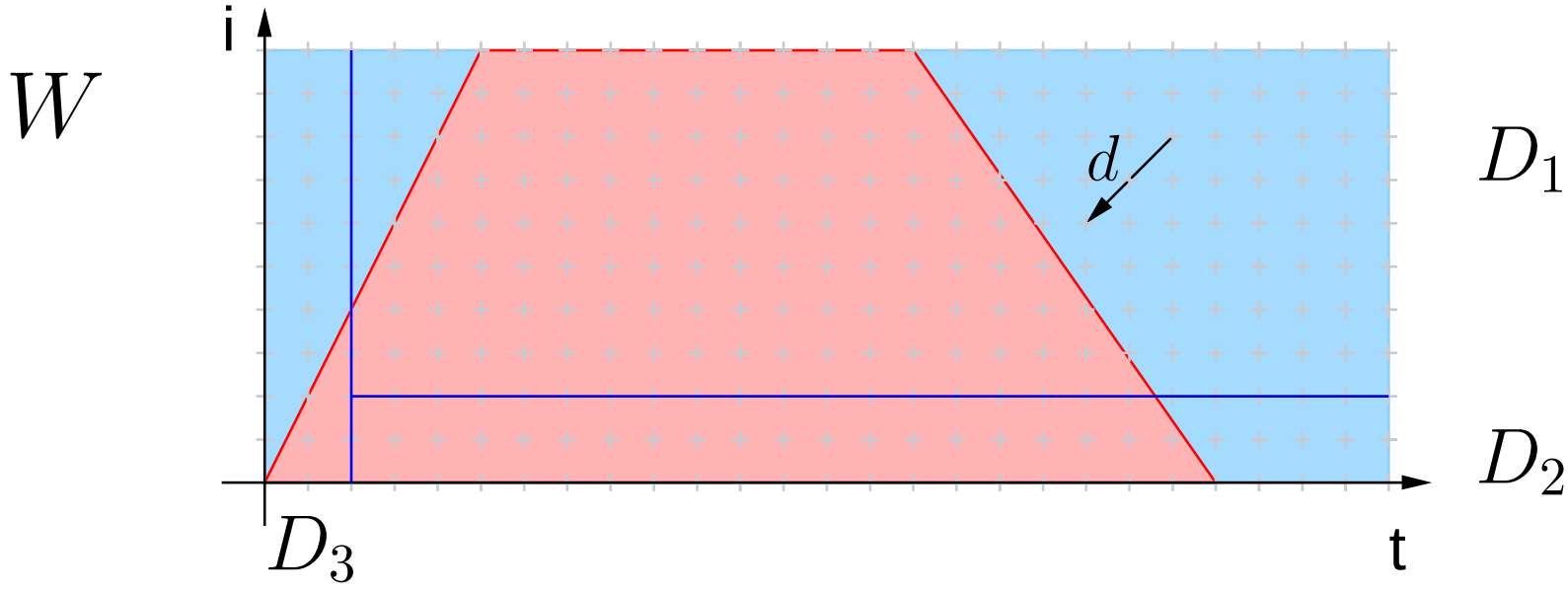
$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_1 : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (2)



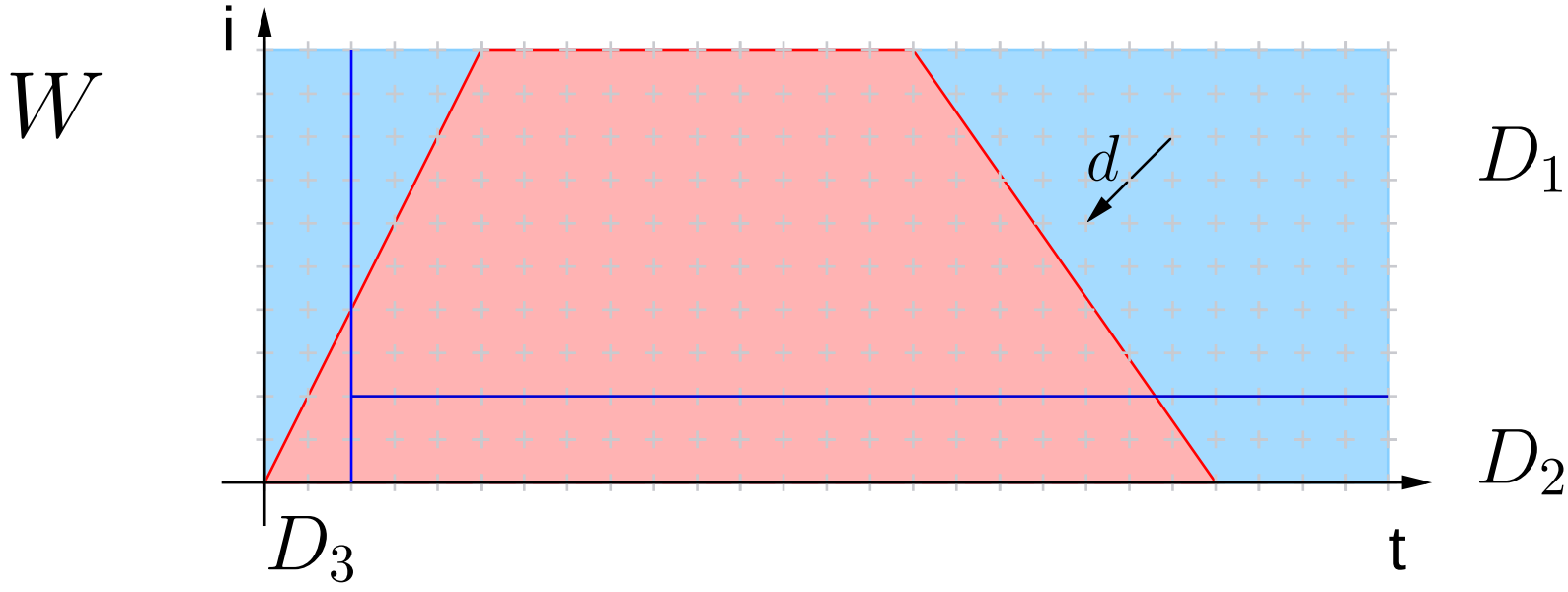
- $\mathcal{D}_{rouge} : W \downarrow tt$

Substitution par constante (2)



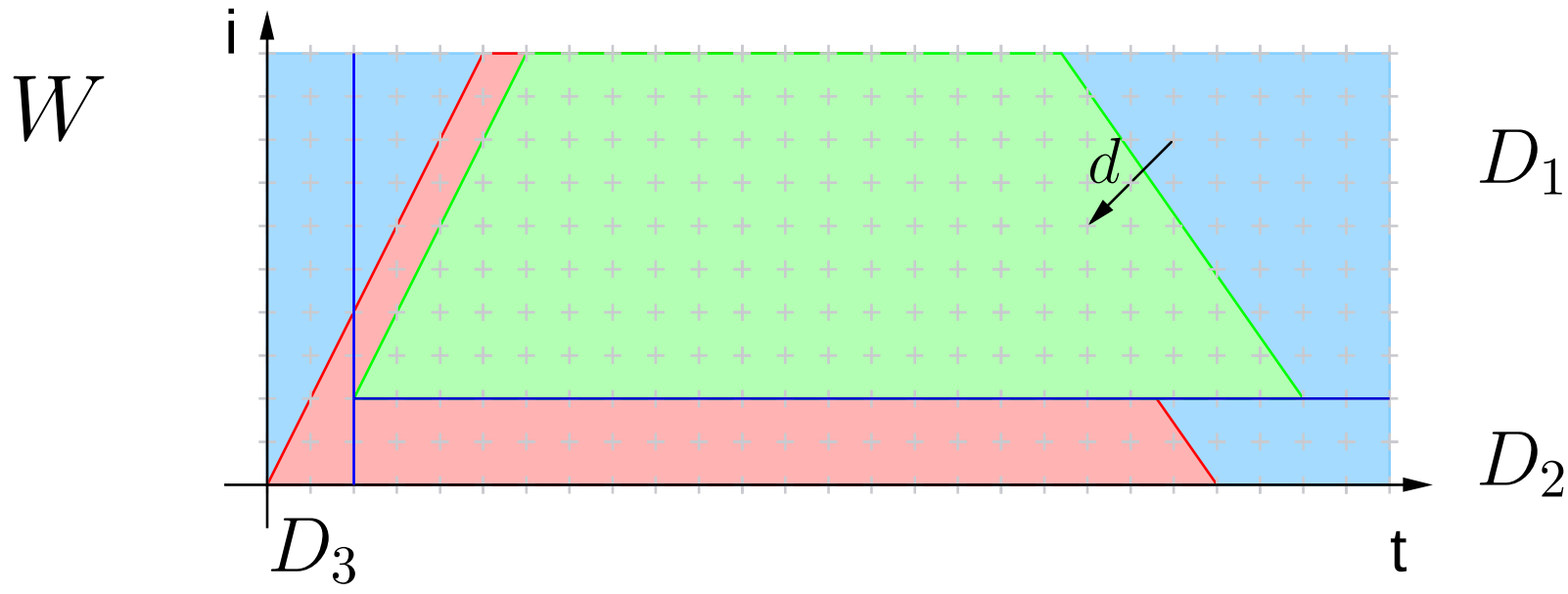
- $\mathcal{D}_{rouge} : W \downarrow tt$
- Supposons $\mathcal{D}_{rouge} : W \downarrow tt$ vraie alors

Substitution par constante (2)



- $\mathcal{D}_{rouge} : W \downarrow tt$
- Supposons $\mathcal{D}_{rouge} : W \downarrow tt$ vraie alors
- Substituer toutes les instances de W dans le domaine $\mathcal{D}_{rouge}$ par *True* dans la partie droite de toutes les équations, en particulier W

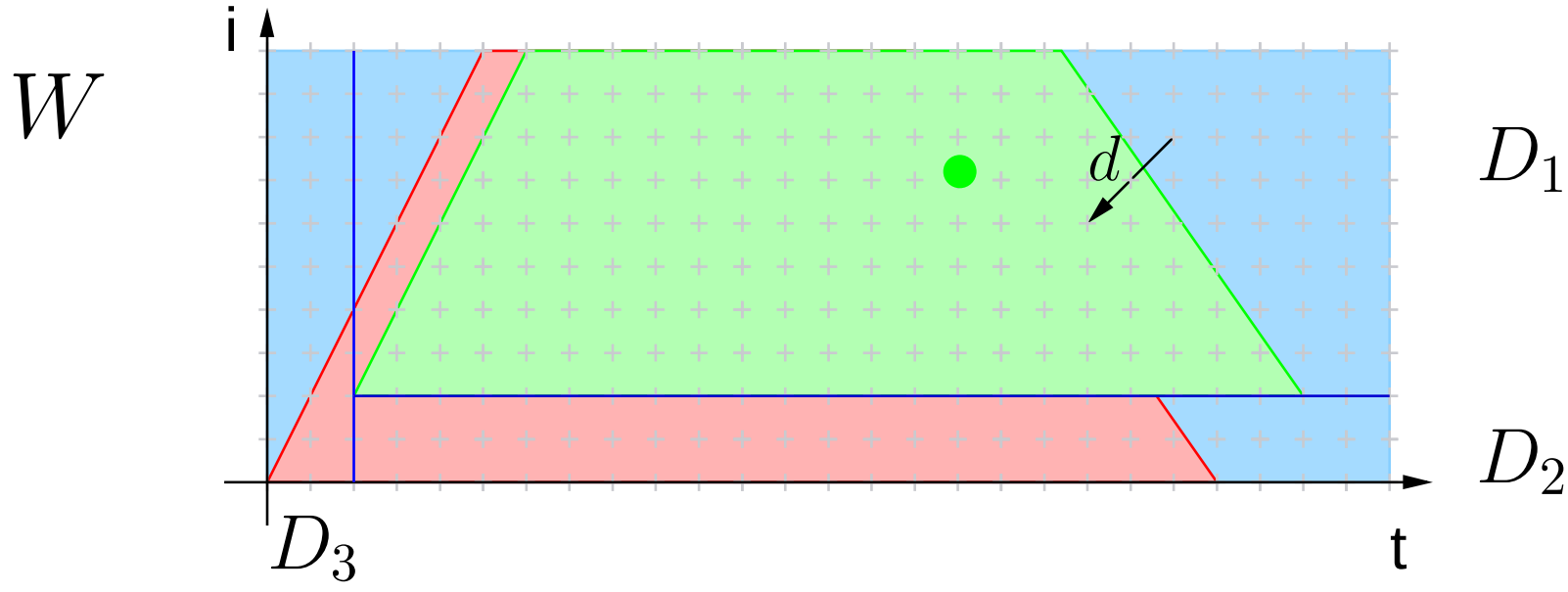
Substitution par constante (3)



$$D_{vert} = D_1 \cap d^{-1}(D_{rouge})$$

$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_1 : W.d \vee Y.\delta \end{cases}$$

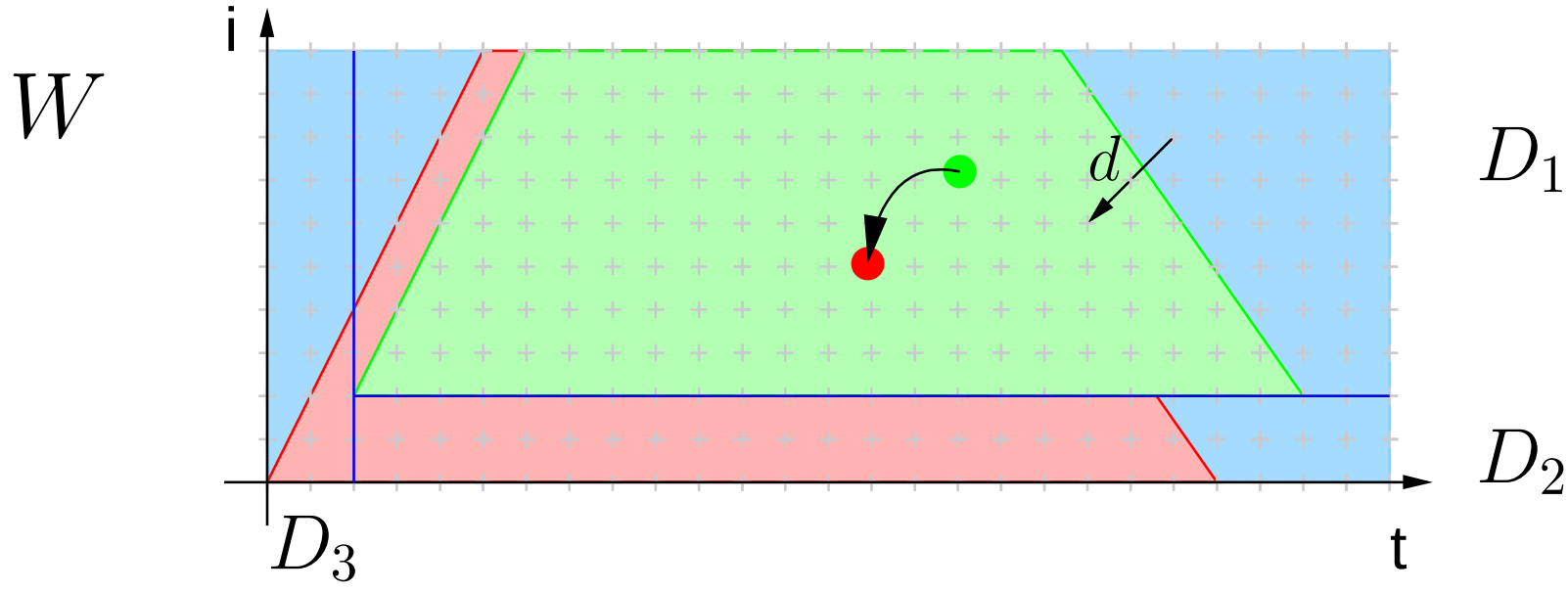
Substitution par constante (3)



$$D_{vert} = D_1 \cap d^{-1}(D_{rouge})$$

$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_1 : W.d \vee Y.\delta \end{cases}$$

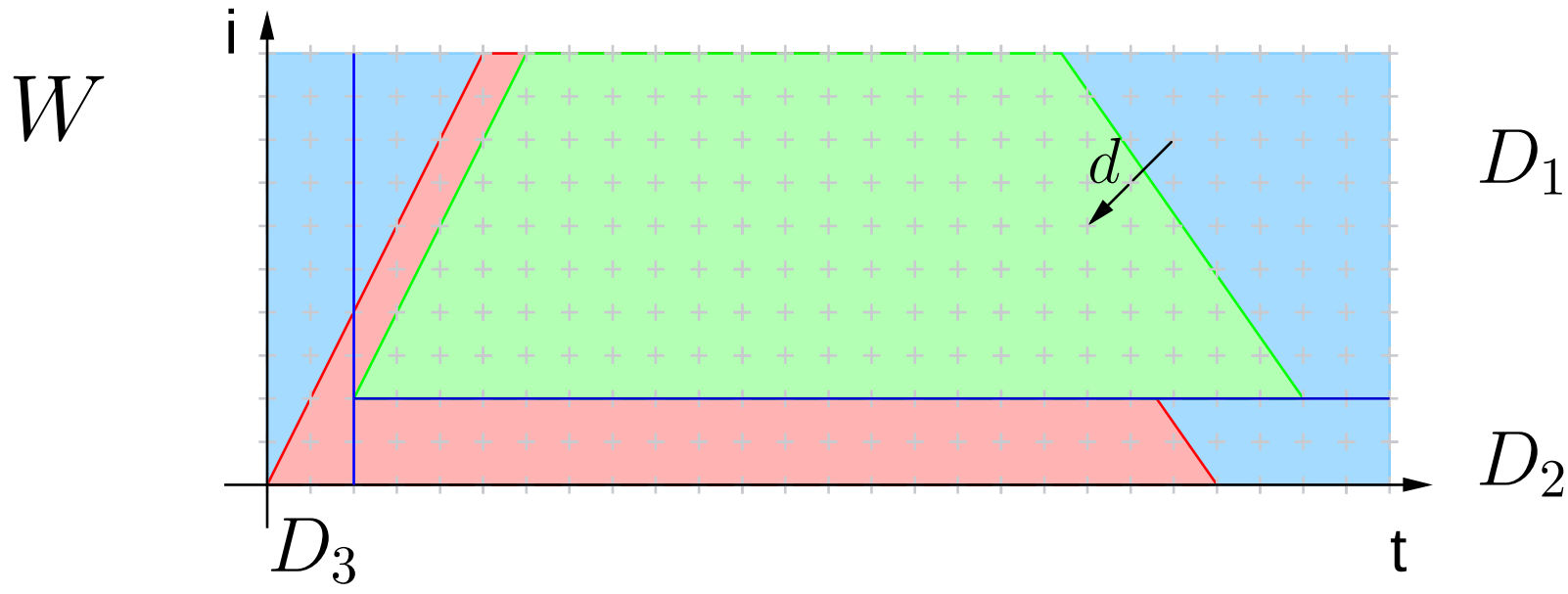
Substitution par constante (3)



$$D_{vert} = D_1 \cap d^{-1}(D_{rouge})$$

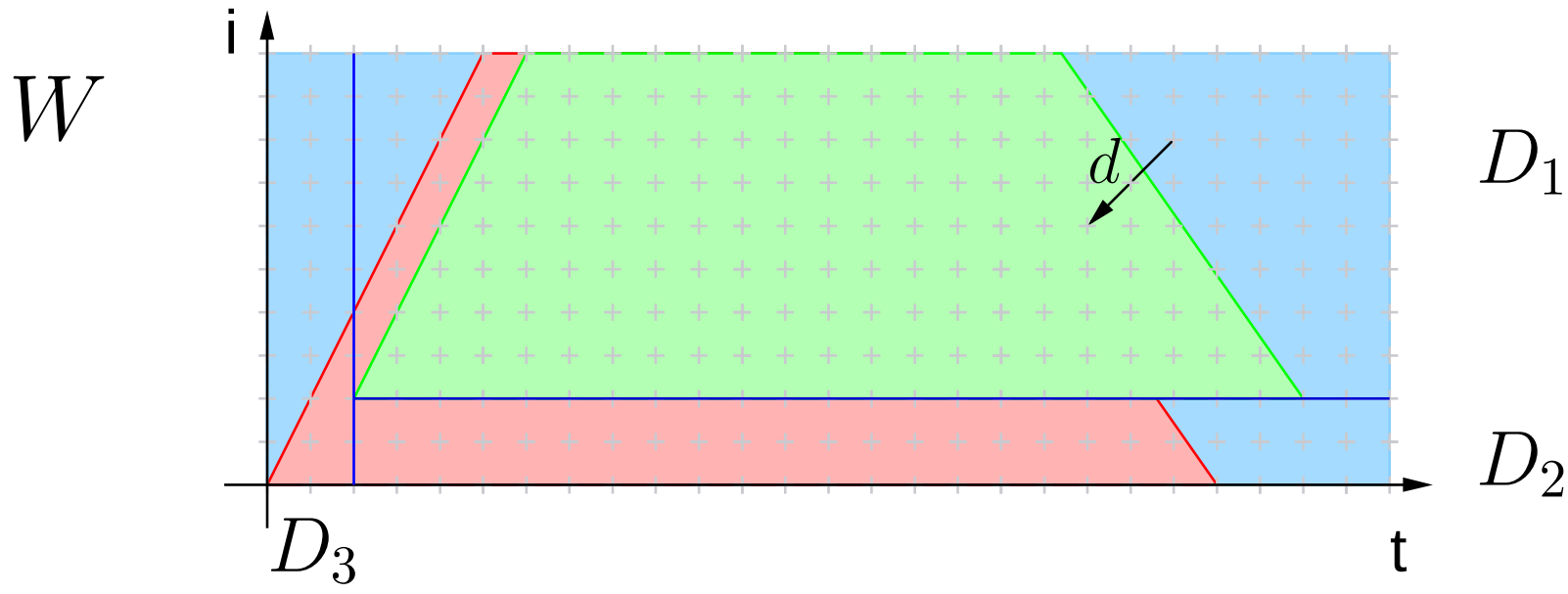
$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_1 : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (3)



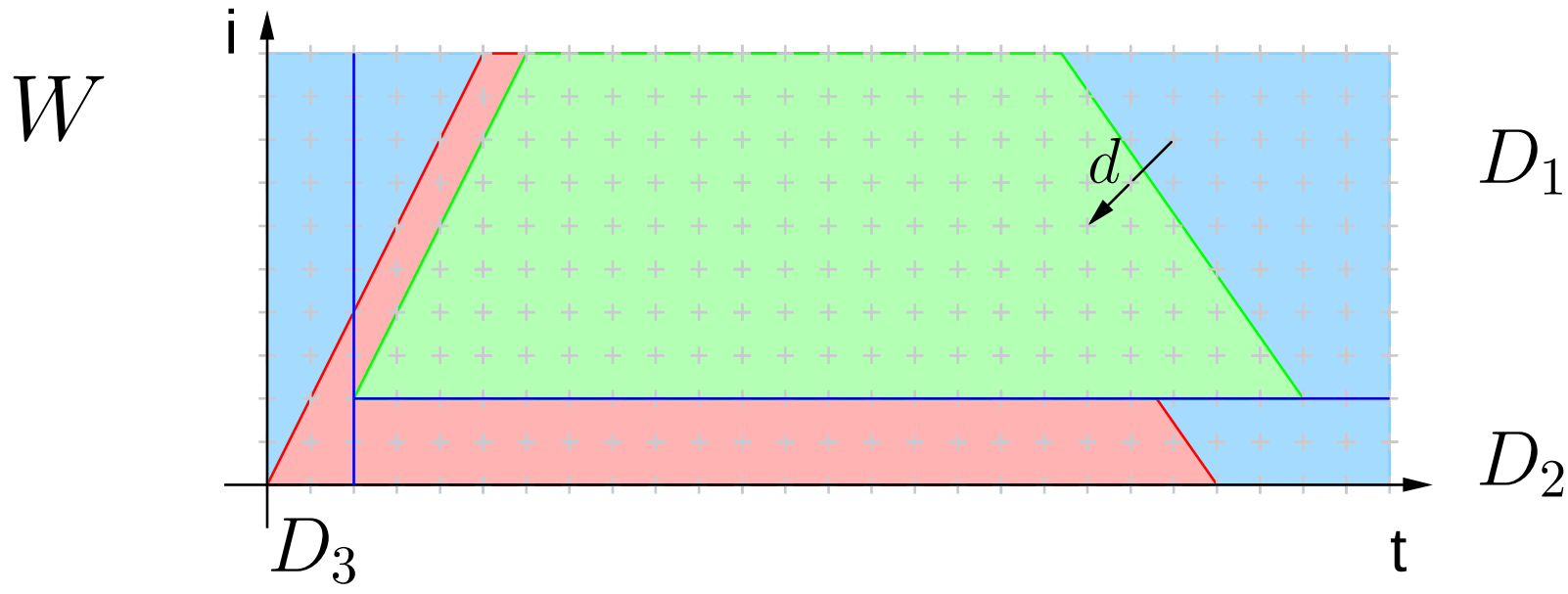
$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_{vert} : W.d \vee Y.\delta \\ D_1 \setminus D_{vert} : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (3)



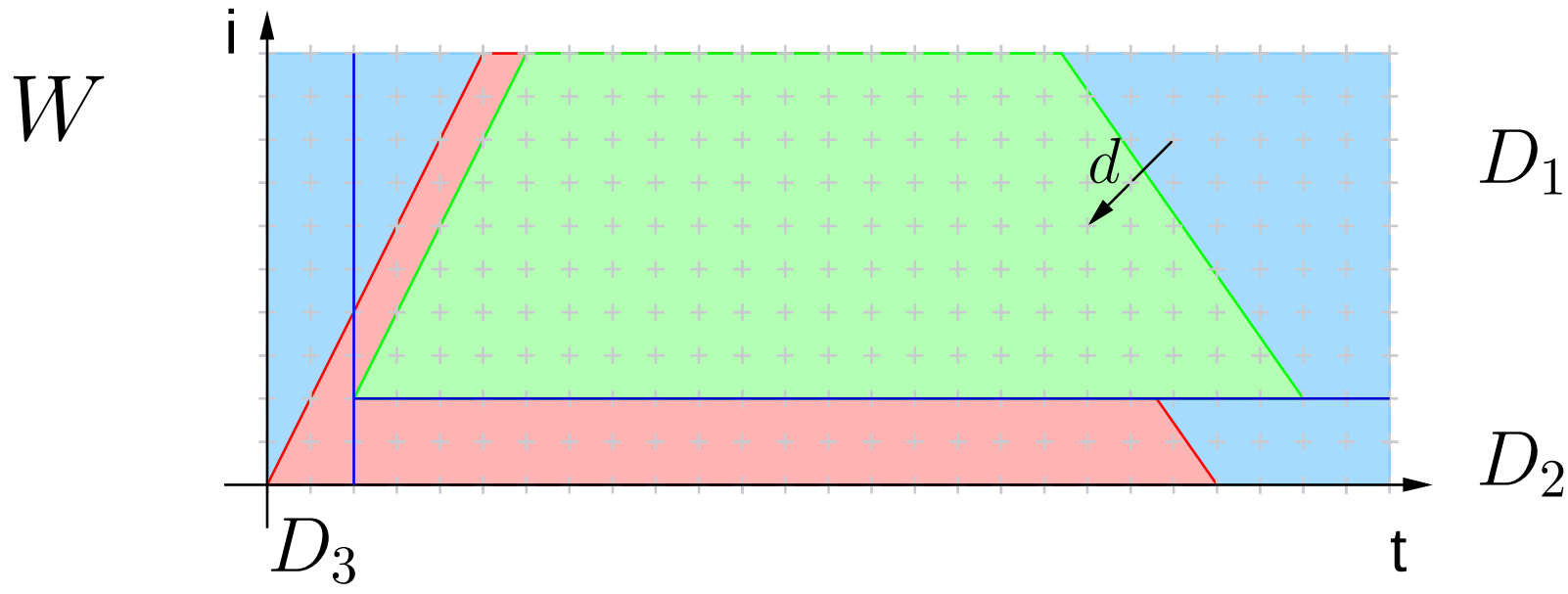
$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_{vert} : \mathit{True} \vee Y.\delta \\ D_1 \setminus D_{vert} : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (3)



$$W_{reec} = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_{vert} : \mathit{True} \\ D_1 \setminus D_{vert} : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (3)



$$W = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_1 : W.d \vee Y.\delta \end{cases} \Leftrightarrow W_{reec} = \begin{cases} D_2 : Y.\delta \\ D_3 : \mathit{True} \\ D_{vert} : \mathit{True} \\ D_1 \setminus D_{vert} : W.d \vee Y.\delta \end{cases}$$

Substitution par constante (4)

$$\frac{S \vdash \mathcal{D} : W_{reec} \downarrow v}{S \vdash \mathcal{D} : W \downarrow v}\ \text{SUBS}$$

- THÉORÈME : La règle de substitution est sûre par rapport à la sémantique dénotationnelle
- Démonstration : Par récurrence sur l'indice temporel
  une étape d'induction $=$ substitution d'un ensemble d'instances définies à un instant t.

Motifs et auto-dépendances

- Règle de recherche de motifs :
  - substitution d'une expression par une variable
    $M = X \vee Y, \quad Z = X.d \vee Y.d \vee W \rightsquigarrow Z = M.d \vee W$
  - résolution d'un système d'équations linéaires à valeur dans $\mathbb{Z}$ pour calculer les dépendances
- Règle de recherche d'auto-dépendances :
  - faire apparaître des auto-dépendances dans une variable définie sans auto-dépendance :
    $W = X.d_1 \vee X.d_2 \quad \rightsquigarrow \quad W = W.\delta \vee \ldots$
  - utilisation de la recherche de motifs

Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

Algorithme

- Alternance des règles de preuve
- Construction d'un arbre de preuve
- Un nœud de l'arbre = une formule
- Choix de la règle à appliquer en fonction de la structure de la formule du nœud
- Arrêt de la construction :
  - soit une feuille fermée : on peut appliquer un axiome
  - soit une feuille pendante : aucune règle ne s'applique ou formule déjà rencontrée

Terminaison de l'algorithme

- THÉORÈME : La construction de l'arbre est finie
- Preuve : nouvelles formules
  - générées par la recherche d'auto-dépendance (substitution de variables par leur définition)
  - théorème caractérisant les cas où l'itération de la substitution par constante et de la recherche d'auto-dépendance est finie

Arbre de preuve

$$X = \begin{cases} \mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3) \\ \mathcal{D}_2 : (Y.d_5 \wedge Z.d_6) \vee Y.d_4 \end{cases}$$

Arbre de preuve

$$X = \begin{cases} \mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3) \\ \mathcal{D}_2 : (Y.d_5 \wedge Z.d_6) \vee Y.d_4 \end{cases}$$

SEP

$\mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3)$ $\qquad$ $\mathcal{D}_2 : (Y.d_5 \vee Z.d_6) \wedge Y.d_4$

Arbre de preuve

$$X = \begin{cases} \mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3) \\ \mathcal{D}_2 : (Y.d_5 \wedge Z.d_6) \vee Y.d_4 \end{cases}$$

SEP

$\mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3)$ $\qquad$ $\mathcal{D}_2 : (Y.d_5 \vee Z.d_6) \wedge Y.d_4$

ET

$\mathcal{D}_1 : X.d_1$ $\qquad$ $\mathcal{D}_1 : Y.d_2 \vee Y.d_3$

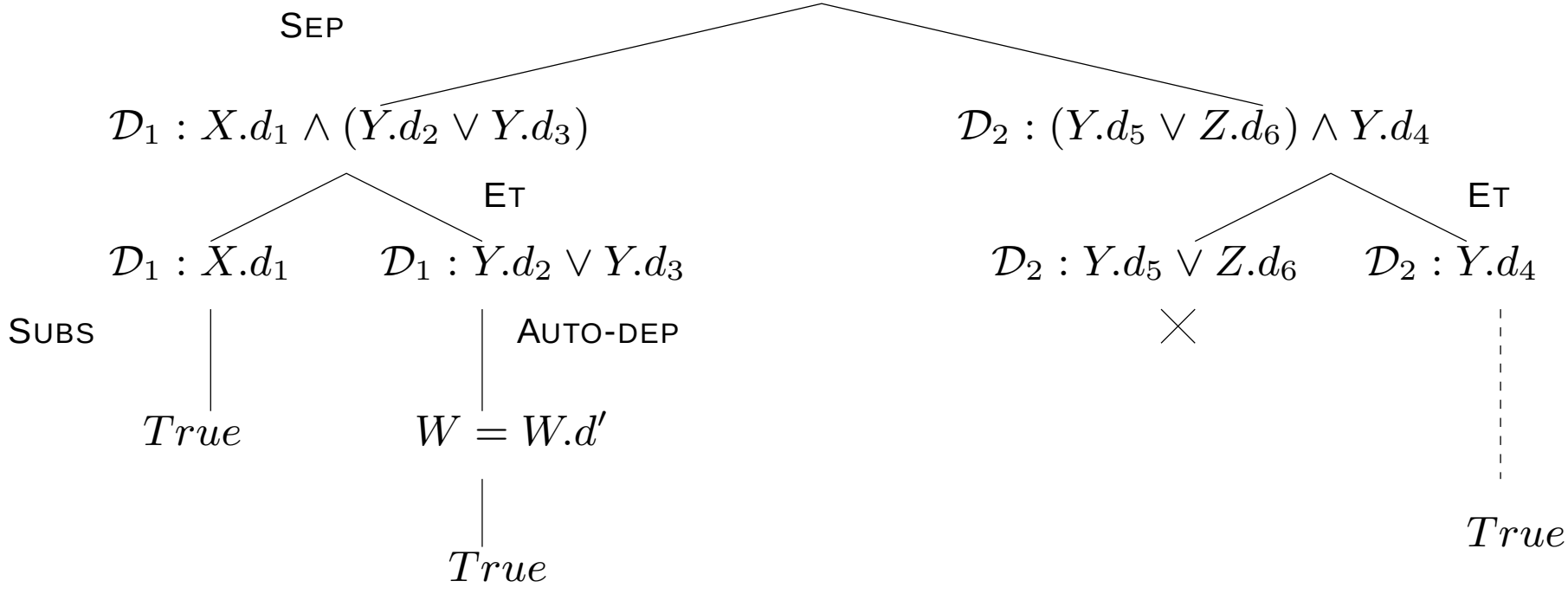
SUBS $\qquad$ AUTO-DEP

$True$ $\qquad$ $W = W.d'$

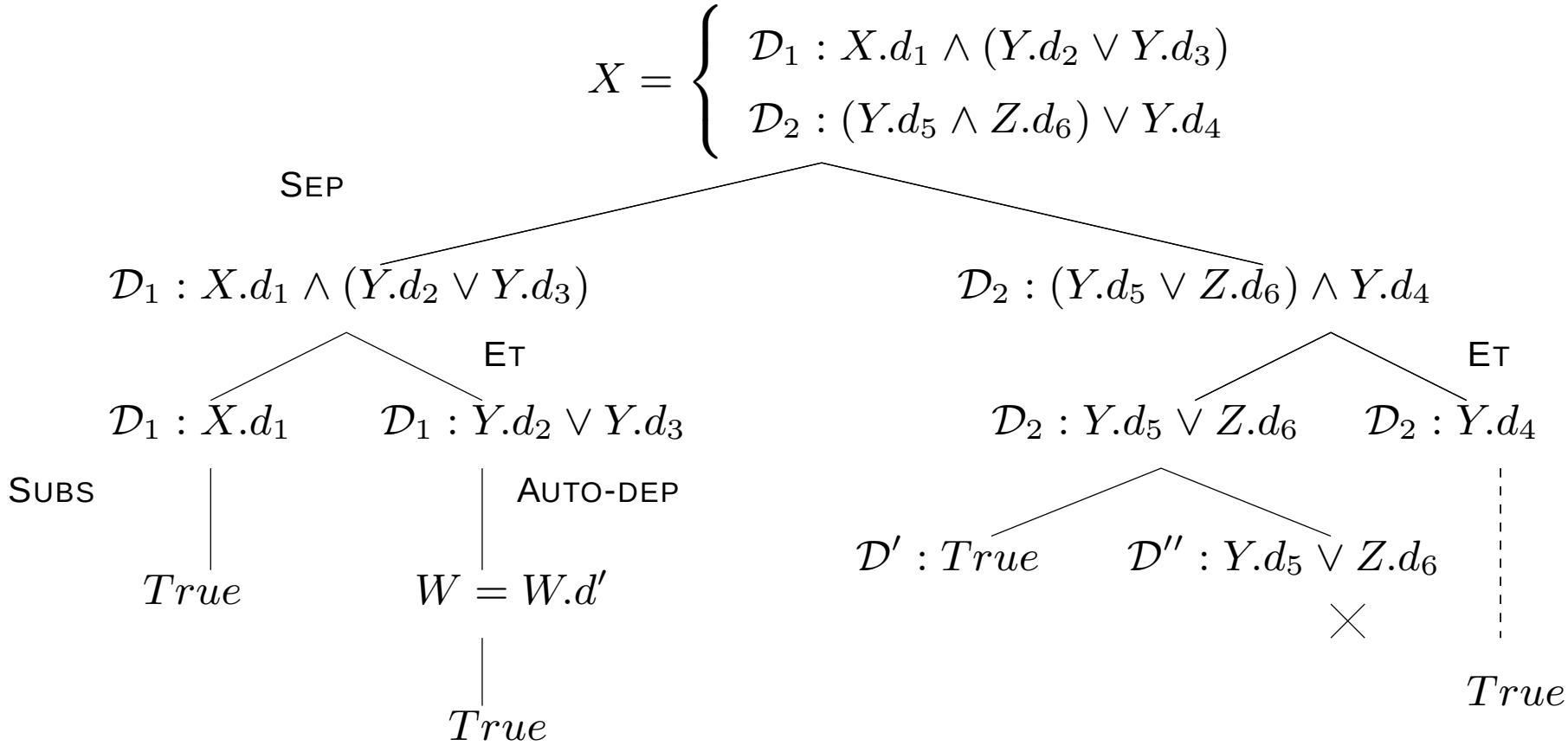
$True$

Arbre de preuve

$$X = \begin{cases} \mathcal{D}_1 : X.d_1 \wedge (Y.d_2 \vee Y.d_3) \\ \mathcal{D}_2 : (Y.d_5 \wedge Z.d_6) \vee Y.d_4 \end{cases}$$



Arbre de preuve

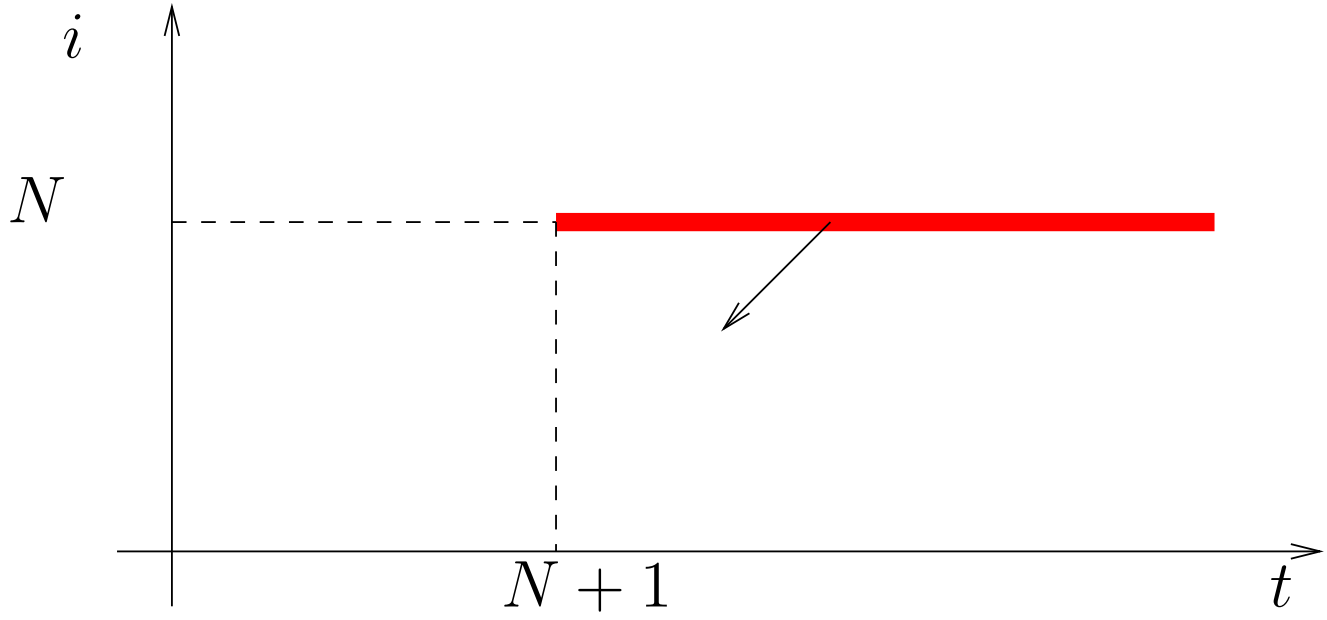


Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

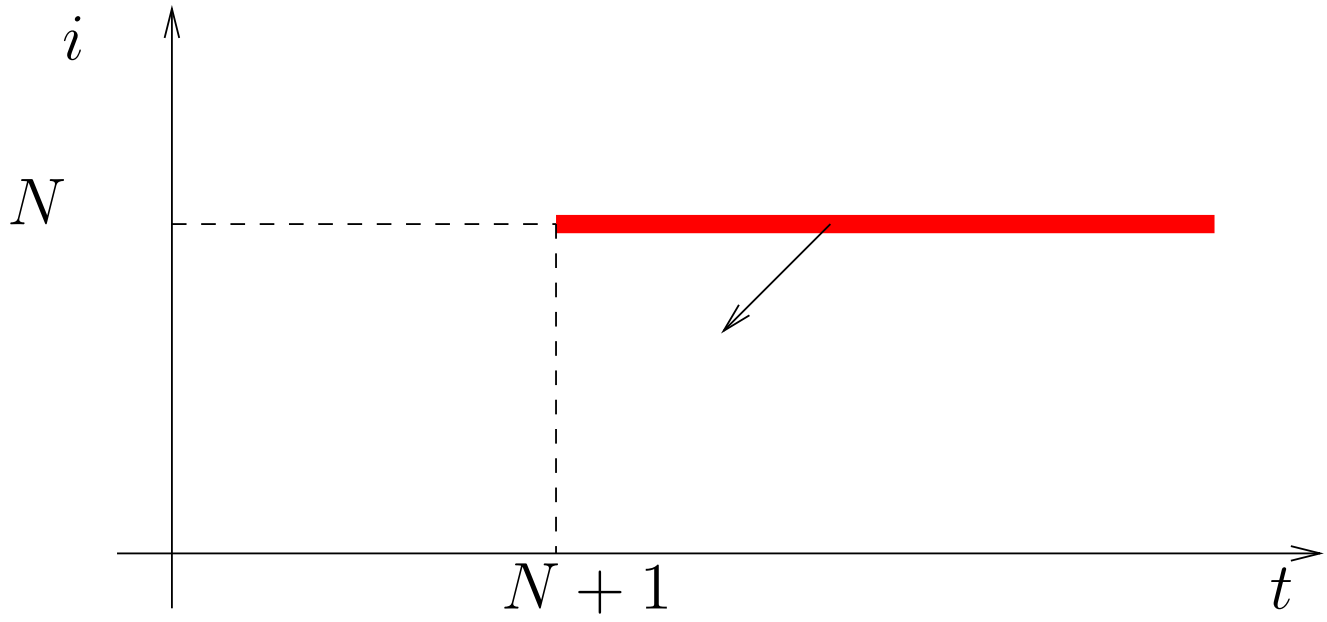
Agrandissement

- On veut prouver : $\{t, i | N \leq t \leq 3N, i = N\} : A \downarrow tt$



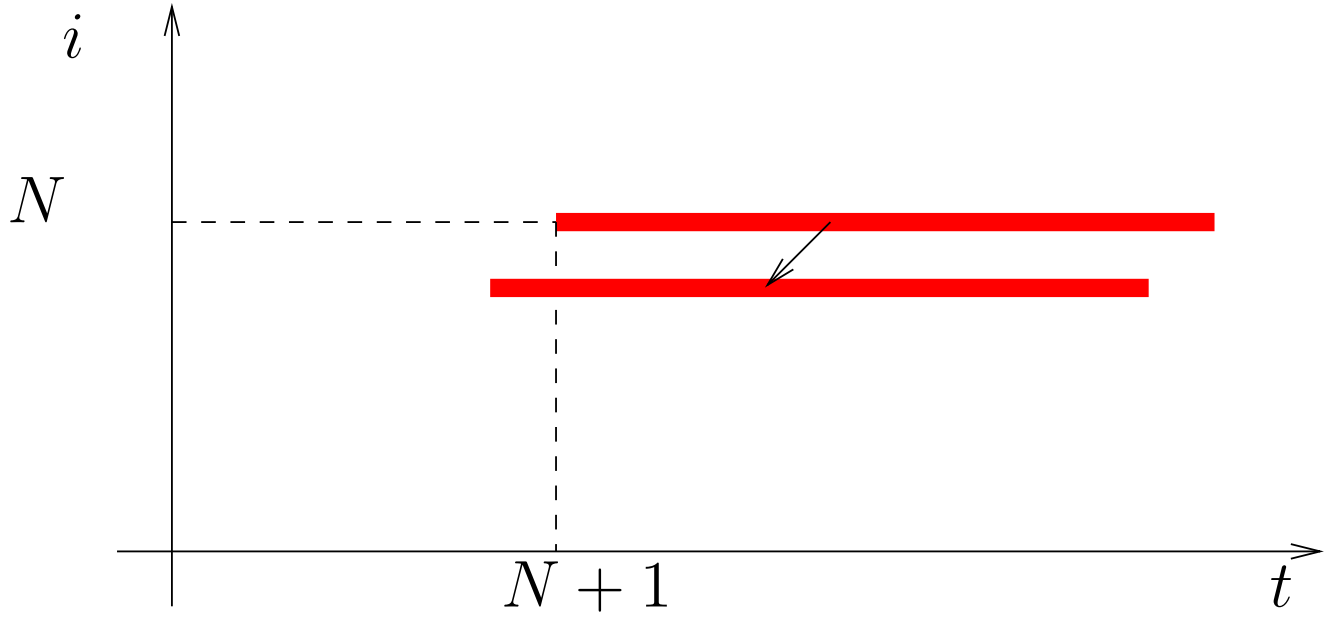
Agrandissement

- On veut prouver : $\{t, i | N \leq t \leq 3N, i = N\} : A \downarrow tt$
- $A = \text{if empty}[t, i] \text{ then } C[t-1, i] \text{ else } A[t-1, i-1]$



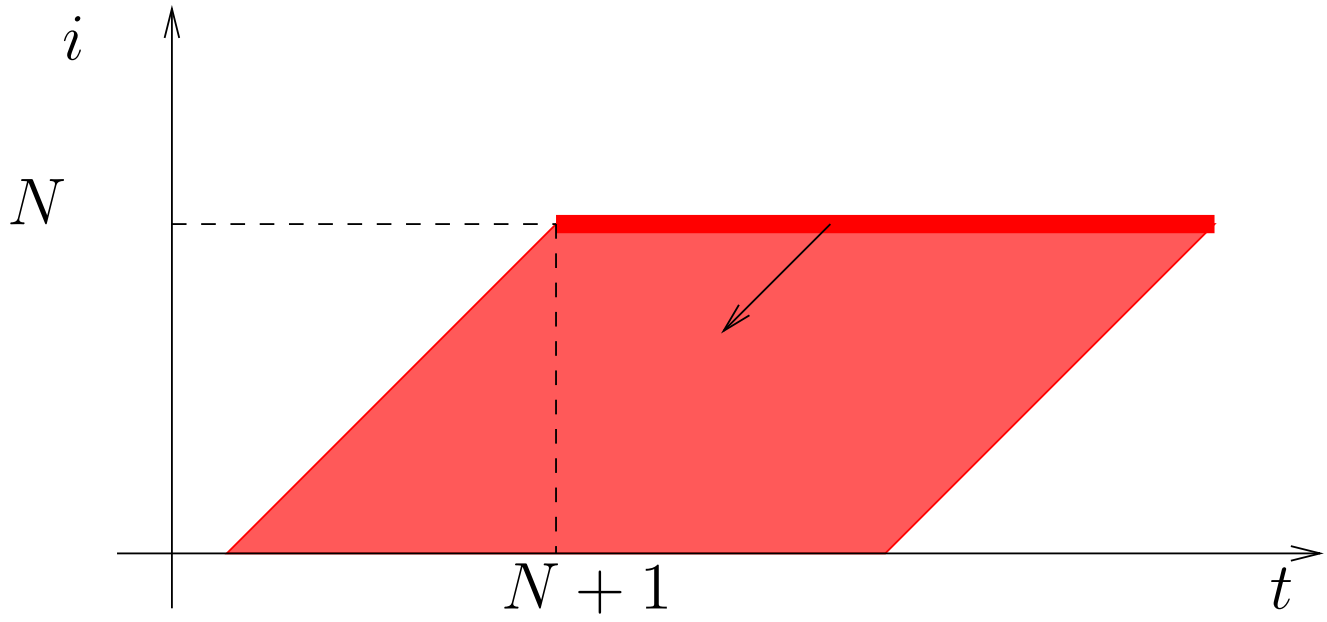
Agrandissement

- On veut prouver : $\{t, i | N \leq t \leq 3N, i = N\} : A \downarrow tt$
- $A = \text{if empty}[t, i] \text{ then } C[t-1, i] \text{ else } A[t-1, i-1]$
- Valeur de $A[t, i]$ dépend de $A[t-1, i-1]$



Agrandissement

- On veut prouver : $\{t, i | N \leq t \leq 3N, i = N\} : A \downarrow tt$
- $A = \text{if empty}[t, i] \text{ then } C[t-1, i] \text{ else } A[t-1, i-1]$
- Valeur de $A[t, i]$ dépend de $A[t-1, i-1]$
- Agrandir dans la direction de l'auto-dépendance



Agrandissement : pipelines

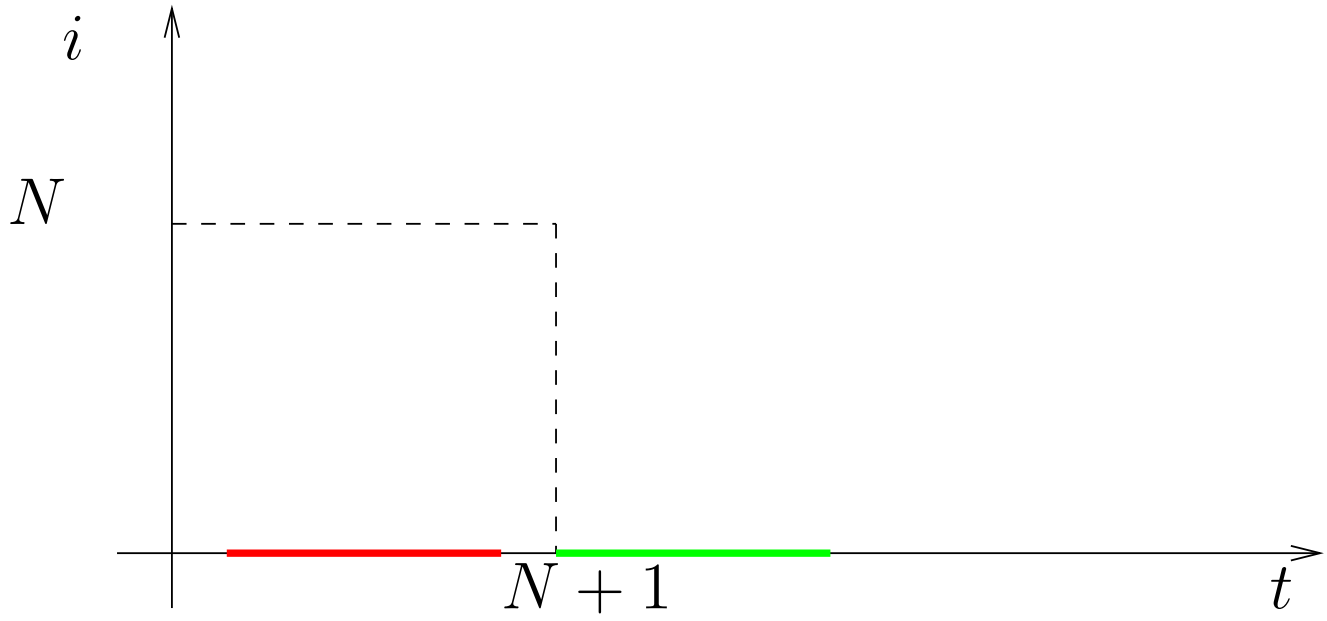
- étude des variables pipelinées :

$$empty = \begin{cases} \{t, i \mid t \leq N\} : False; \\ \{t, i \mid N+1 \leq t \leq 2N; i = 1\} : True \\ \{t, i \mid t > N; 1 < i \leq N\} : empty.(t, i \rightarrow t-2, i-1) \end{cases}$$

Agrandissement : pipelines

- étude des variables pipelinées :

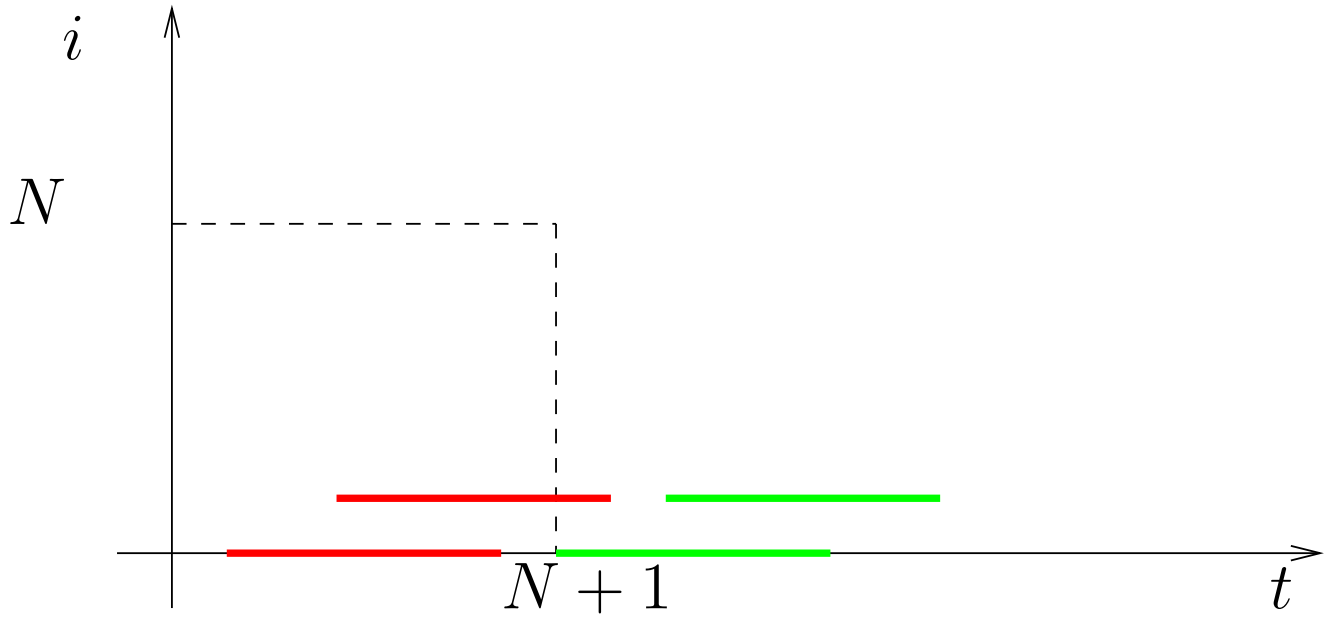
$$empty = \begin{cases} \{t, i \mid t \leq N\} : False; \\ \{t, i \mid N+1 \leq t \leq 2N; i = 1\} : True \\ \{t, i \mid t > N; 1 < i \leq N\} : empty.(t, i \rightarrow t-2, i-1) \end{cases}$$



Agrandissement : pipelines

- étude des variables pipelinées :

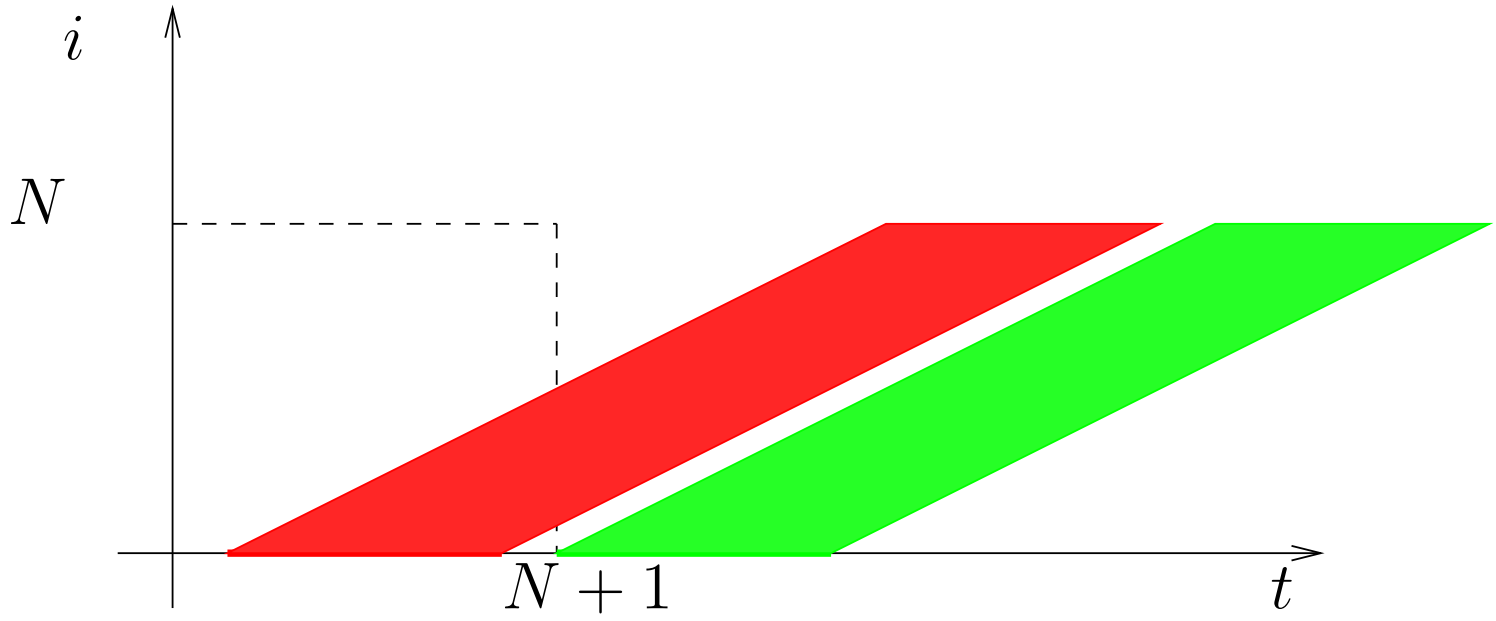
$$empty = \begin{cases} \{t, i \mid t \leq N\} : False; \\ \{t, i \mid N+1 \leq t \leq 2N; i = 1\} : True \\ \{t, i \mid t > N; 1 < i \leq N\} : empty.(t, i \rightarrow t-2, i-1) \end{cases}$$



Agrandissement : pipelines

- étude des variables pipelinées :

$$empty = \begin{cases} \{t, i \mid t \leq N\} : False; \\ \{t, i \mid N+1 \leq t \leq 2N; i = 1\} : True \\ \{t, i \mid t > N; 1 < i \leq N\} : empty.(t, i \rightarrow t-2, i-1) \end{cases}$$



Agrandissement

- substitution de $empty$ par sa valeur

$$A = \text{if } \text{empty}[t, i] \text{ then } C[t-1, i] \text{ else } A[t-1, i-1]$$

Agrandissement

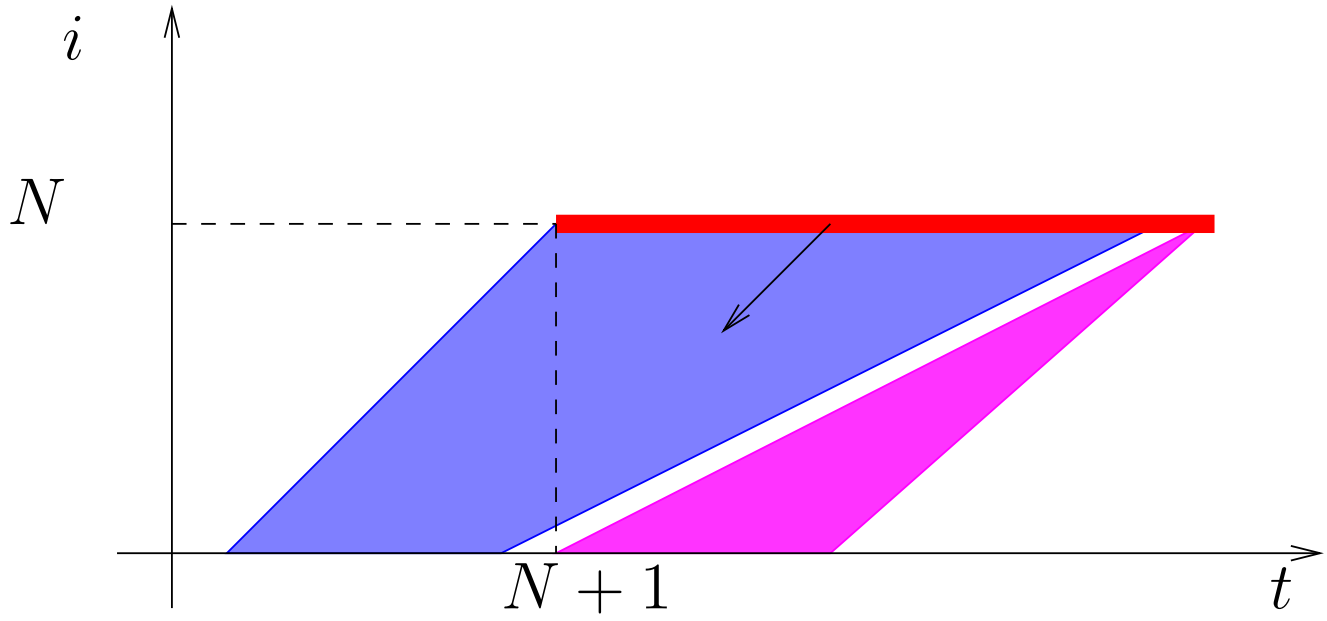
- substitution de $empty$ par sa valeur

$$A = \begin{cases} \{t, i \mid 2i + N \leq t \leq i + 2N; 1 \leq i \leq N\} : C.(t, i \rightarrow t - 1, i) \\ \{t, i \mid i \leq t < 2i + N; 1 \leq i \leq N\} : A.(t, i \rightarrow t - 1, i - 1); \end{cases}$$

Agrandissement

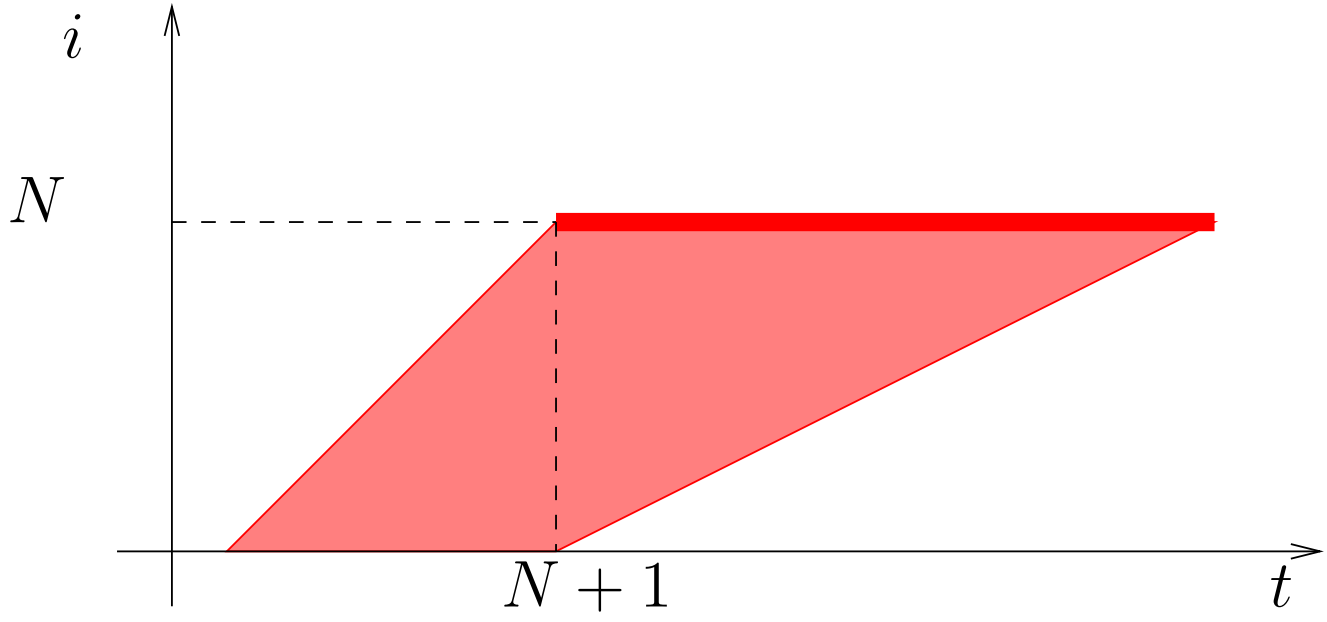
- substitution de $empty$ par sa valeur

$$A = \begin{cases} \{t, i \mid 2i + N \leq t \leq i + 2N; 1 \leq i \leq N\} : C.(t, i \rightarrow t - 1, i) \\ \{t, i \mid i \leq t < 2i + N; 1 \leq i \leq N\} : A.(t, i \rightarrow t - 1, i - 1) \end{cases}$$



Agrandissement

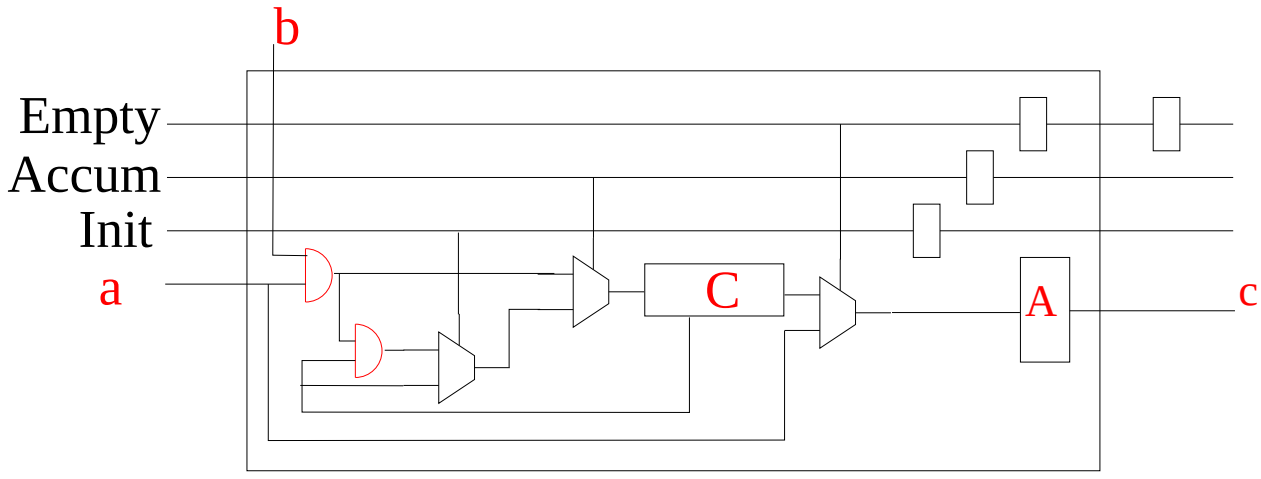
- substitution de $empty$ par sa valeur
- agrandir dans le sens des dépendances :



Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

Exemple : une cellule



- Entrées a et b: suite de N valeurs “vrai” suivies par N valeurs “faux”
- Prouver que c vaut toujours vrai.
- Preuve effectuée en 2 parties : périodicité et preuve sur une période

Preuve sur la première période

- c vaut toujours vrai

$$c = A.(t \rightarrow t, N)$$

Preuve sur la première période

- c vaut toujours vrai

$$c = A.(t \rightarrow t, N)$$

- Agrandir le domaine de A

Preuve sur la première période

- c vaut toujours vrai

$$c = A.(t \rightarrow t, N)$$

- Agrandir le domaine de A
- Générer différentes obligations de preuve sur différentes variables

Preuve sur la première période

- c vaut toujours vrai

$$c = A.(t \rightarrow t, N)$$

- Agrandir le domaine de A
- Générer différentes obligations de preuve sur différentes variables
- Pour une des variables C, agrandir le domaine

Preuve sur la première période

- c vaut toujours vrai

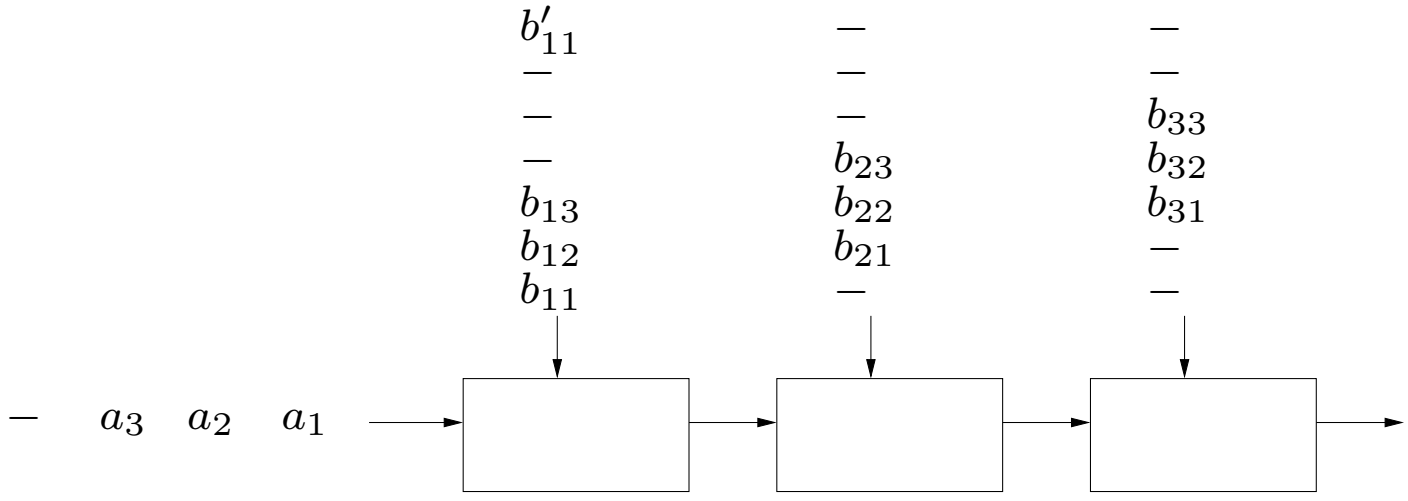
$$c = A.(t \rightarrow t, N)$$

- Agrandir le domaine de A
- Générer différentes obligations de preuve sur différentes variables
- Pour une des variables C, agrandir le domaine
- Puis toutes les preuves réussissent

Résultats

c vaut toujours vrai équivalent à 2 conditions :

- $\forall t \in \{t | 1 \leq t \leq N\}$, $a[t]$ vaut vrai
- $\forall t, i \in \{t, i | i + 1 \leq t \leq N + i; 1 \leq i \leq N\}$, $b[t, i]$ vaut vrai



Conditions : conséquence de la spécification

Plan

- Exemple introductif
- Modèle polyédrique
- Logique polyédrique
- Construction d’un arbre de preuve
- Utilisation d’un opérateur d’agrandissement
- Illustration
- Conclusion

Conclusion

- Prouver des propriétés de contrôle sur des systèmes paramétrés décrits dans le modèle polyédrique
- Logique polyédrique
  - règles classiques
  - règle de substitution par constante
  - règles de recherche de motifs et recherche d'auto-dépendance
- Un algorithme de preuve
- Heuristiques d'agrandissement

Autres travaux

- Implémentation (prototype)
- Étude d'un filtre adaptatif
- Enrichissement de la logique polyédrique :
  - conjonctions, disjonctions,. . . de formules
  - prise en compte des hypothèses
  - ajout d'un contexte
- Développement de tactiques de preuves : utilisation des pseudo-pipelines et ajout de paramètres, récurrence, absurde
- Exemple de référence : un arbitre matériel (exclusion mutuelle, priorité)

Perspectives

- Optimiser le prototype
- Exprimer des propriétés quantifiées universellement et existentiellement (ajout de paramètres pour représenter les indices quantifiés existentiellement)
- Travailler sur des $\mathbb{Z}$-polyèdres pour exprimer par exemple des propriétés valides pour les instants pairs
- Utilisation de treillis de polyèdres : des résultats sur les systèmes d'additions de vecteurs avec états (Reutenauer)
- Propriétés de sûreté sur des signaux entiers (utilisation des polyèdres pour représenter l'espace des valeurs ?)
- Propriétés de vivacité

Merci de votre attention

Domaine vide

- Polylib : Calculs dans $\mathbb{Q}^n$
- MMAlpha : $< \rightsquigarrow \leq$
  $a < b \quad \rightsquigarrow \quad a + 1 \leq b$
- Polyèdre $\{t \mid 0 < t < 1\}$ vide :
  - $\{t \mid 1 \leq t \leq 0\} \quad \rightsquigarrow$ vide dans $\mathbb{Q}$
- Polyèdre $\{t \mid 0 < 10t < 10\}$ non vide :
  - $\{t \mid 1 \leq 10t \leq 9\} \quad \rightsquigarrow$ non vide dans $\mathbb{Q}$
- Solutions : utiliser omega, PIP, polynômes d'Ehrhardt
  ...