Horloges 1-synchrones sous-spécifiées et non-déterminisme

Guillaume Iooss, Albert Cohen, Marc Pouzet

ENS - PARKAS

February 1, 2019

Background - Langage synchrone

- Manipule des flots infinis de valeurs
- Production des valeurs synchronisées par un tick global
- Opérations sont point-à-point sur les valeurs
- Possible d'accéder à des valeurs passées ("fby" $\approx$ mémoire)

x	0	1	1	2	...
y	4	-2	1	4	...
42	42	42	42	42	...
$x + y$	4	-1	2	6	...
42 fby y	42	4	-2	1	...

Background - Horloges

- Un flux peut ne pas avoir de valeur sur certains ticks
- **Horloge**: $x :: clk$
  - Signale la présence d'une valeur
  - Flux de booléen associé à un flux de valeur
  - Peut être une variable booléen de valeur arbitraire
- Op temporels: échantillonnage (when) et fusion (merge)
- Vérification de cohérence des horloges: analyse de "clocking"

x	$:: c$	0	1	1	2	...
b	$:: c$	t	f	t	t	...
$z = x$ when b	$:: c$ on b	0	–	1	2	...
y	$:: c$ on not b	–	42	–	–	...
merge b z y	$:: c$	0	42	1	2	...

Background - Lustre

- Langage équationnel pour les programmes synchrones (Autre langage proche: Scade, Heptagon, . . .)

```
node accumulator(i : int) returns (o : int)
var x : int
let
   x = 0 fby o;
   o = x + i;
tel
```

Background - Lustre

- Langage équationnel pour les programmes synchrones (Autre langage proche: Scade, Heptagon, . . .)

```
node accumulator(i : int) returns (o : int)
var x : int
let
   x = 0 fby o;
   o = x + i;
tel
```

- **Génération de code:**
  - Fonctions "reset" et "step"
  - Boucle "while" infinie (1 itération = 1 tick de base)
  - Horloges: entourer par des "if"

Background - Modèle N-synchrone

- **Modèle N-synchrone**:
  - Horloges ultimement périodiques
  - Exemple: 101(1001)
  - Strictement périodique: pas de phase d'initialisation

$\Rightarrow$ Système d'horloge devient plus prédictible

Background - Modèle N-synchrone

- **Modèle N-synchrone**:
  - Horloges ultimement périodiques
  - Exemple: 101(1001)
  - Strictement périodique: pas de phase d'initialisation

$\Rightarrow$ Système d'horloge devient plus prédictible

- **buffer**: Communication entre variables sur deux horloges différentes
  - Horloges doivent être compatible (adaptabilité: $<:$)

  $\Rightarrow$ Peut calculer la taille d'un buffer

Horloges 1-synchrones

- **Cas d'étude:** programmes d'intégration
  Orchestre toutes les tâches de l'application
  - Plusieurs périodes harmoniques (ex: 5 / 10 / 20 ms)
  - Tâches n'apparaissent qu'une fois par période

Horloges 1-synchrones

- **Cas d'étude:** programmes d'intégration
  Orchestre toutes les tâches de l'application
  - Plusieurs périodes harmoniques (ex: 5 / 10 / 20 ms)
  - Tâches n'apparaissent qu'une fois par période

- **Horloge 1-synchrone:** "$(0^k 10^{n-k-1})$" (ou "$0^k(10^{n-1})$")
  avec $0 \leq k < n$, $n =$ période et $k =$ phase

Horloges 1-synchrones

- **Cas d'étude:** programmes d'intégration
  Orchestre toutes les tâches de l'application
  - Plusieurs périodes harmoniques (ex: 5 / 10 / 20 ms)
  - Tâches n'apparaissent qu'une fois par période

- **Horloge 1-synchrone:** "$(0^k 10^{n-k-1})$" (ou "$0^k(10^{n-1})$")
  avec $0 \leq k < n$, $n =$ période et $k =$ phase

- Programme d'intégration: que des horloges 1-synchrones
  $\rightsquigarrow$ Peut exploiter cette hypothèse dans le compilateur

Dans cette présentation...

Trois modifications incrémentales de Lustre:

1. Spécialisation du calcul d'horloge aux horloges 1-synchrones

Dans cette présentation...

Trois modifications incrémentales de Lustre:

1. Spécialisation du calcul d'horloge aux horloges 1-synchrones

2. Phase des horloges de certaines variables non définies
   - Même valeurs calculées / timing non préservé
   - Contraintes sur les phases obtenues à partir du clocking
   - Fonction de coût et résolution
   - Utilise la solution pour revenir sur du Lustre complet

Dans cette présentation...

Trois modifications incrémentales de Lustre:

1. Spécialisation du calcul d'horloge aux horloges 1-synchrones

2. Phase des horloges de certaines variables non définies
   - Même valeurs calculées / timing non préservé
   - Contraintes sur les phases obtenues à partir du clocking
   - Fonction de coût et résolution
   - Utilise la solution pour revenir sur du Lustre complet

3. Calculs non-déterministes
   - Flou sur quelle valeur d'un flux est utilisée

   $\rightsquigarrow$ Relâche encore plus les contraintes sur les phases
   - Déterminise programme après choix des phases

Calcul d'horloges 1-synchrones - Même période

- Restriction des horloges à des horloges 1-synchrones.
  $\rightsquigarrow$ Impact sur les opérateurs temporels?

Calcul d'horloges 1-synchrones - Même période

- Restriction des horloges à des horloges 1-synchrones.
  $\rightsquigarrow$ Impact sur les opérateurs temporels?
- (**buffer**: phase non spécifiée $\rightsquigarrow$ plus tard)

Calcul d'horloges 1-synchrones - Même période

- Restriction des horloges à des horloges 1-synchrones.
  $\rightsquigarrow$ Impact sur les opérateurs temporels?

- (**buffer**: phase non spécifiée $\rightsquigarrow$ plus tard)
- **delay:** Incrémente la phase / **delay(d)** $=$ delay^d
  - Ne peut pas dépasser la période (pas d'init)

$$\frac{H \vdash \text{a} :: (0^k 1 0^{n-k-1}) \quad 0 \leq d < n-k}{H \vdash \text{delay(d) a} :: (0^{k+d} 1 0^{n-(k+d)-1})}$$

Calcul d'horloges 1-synchrones - Même période

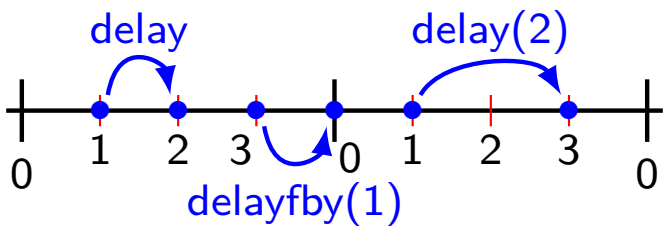
- Restriction des horloges à des horloges 1-synchrones.
  $\rightsquigarrow$ Impact sur les opérateurs temporels?

- (**buffer**: phase non spécifiée $\rightsquigarrow$ plus tard)
- **delay:** Incrémente la phase / **delay(d)** $=$ delay^d
  - Ne peut pas dépasser la période (pas d'init)

$$\frac{H \vdash \text{a} :: (0^k 1 0^{n-k-1}) \quad 0 \leq d < n-k}{H \vdash \text{delay(d) a} :: (0^{k+d} 1 0^{n-(k+d)-1})}$$

- **delayfby(d):** (initialisation / $\approx$ "short fby")

$$\frac{H \vdash a :: (0^k 1 0^{n-k-1}) \quad H \vdash i :: (0^{k+d-n} 1 0^{n-(k+d-n)-1}) \quad 0 \leq k+d-n < n}{H \vdash \text{i delayfby(d) a} :: (0^{k+d-n} 1 0^{n-(k+d-n)-1})}$$



Vers des périodes plus lentes (when)

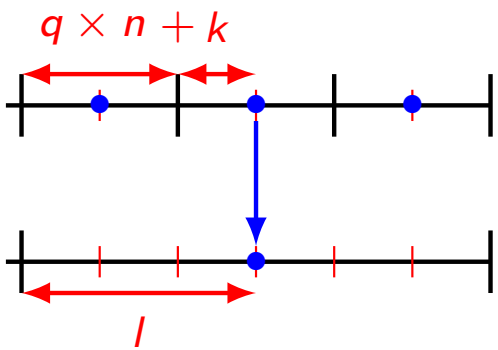
Contraintes sur le calcul d'horloge:

- Périodes doivent être harmoniques
- Argument du when doit être de la forme "$(F^q TF^{p-1-q})$"

Vers des périodes plus lentes (when)

Contraintes sur le calcul d'horloge:

- Périodes doivent être harmoniques
- Argument du when doit être de la forme "$(F^q TF^{p-1-q})$"



y = x when (FTF)

$$\frac{H \vdash a :: (0^k 10^{n-k-1}) \quad m = pn \quad l = qn + k}{H \vdash a \text{ when } (F^q TF^{p-1-q}) :: (0^l 10^{m-l-1})}$$

Vers des périodes plus rapides (merge/current)

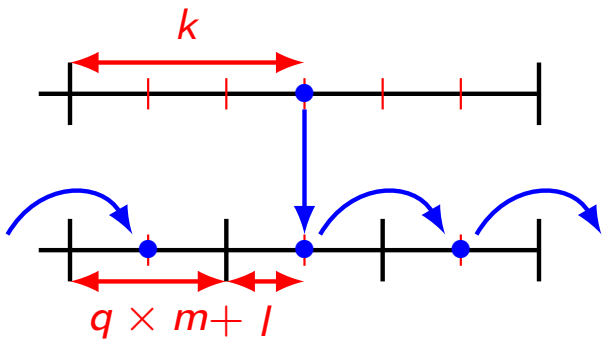
Contraintes sur le calcul d'horloge:

- Périodes doivent être harmoniques
- **merge**: plusieurs branches requises (une par instances)

Vers des périodes plus rapides (merge/current)

Contraintes sur le calcul d'horloge:

- Périodes doivent être harmoniques
- **merge**: plusieurs branches requises (une par instances)
- **current** (répète une valeur, avec une update occasionnelle)
  - Argument (quand l'update arrive) doit être "$(F^q TF^{p-1-q})$"
  - Initialisation ("i")



y = current((FTF), 0, x)

$$\frac{H \vdash a :: (0^k 10^{n-k-1}) \quad H \vdash i :: (0^l 10^{m-l-1}) \quad n = pm \quad l = k - mq}{H \vdash \mathrm{current}((F^q TF^{p-1-q}), i, a) :: (0^l 10^{m-l-1})}$$

Génération de code avec des horloges 1-synchrone

- Exploite la forme des horloges pour générer du code plus efficace
  - Sait exactement quand une activation arrive
  - Tous les "buffer" sont de taille 1 $\rightsquigarrow$ case mémoire

Génération de code avec des horloges 1-synchrone

- Exploite la forme des horloges pour générer du code plus efficace
  - Sait exactement quand une activation arrive
  - Tous les "buffer" sont de taille 1 $\rightsquigarrow$ case mémoire
- Trois possibilités pour la génération de code:
  - Fonction "step" classique (horloge de base)
    - "If" conditions
  - Une fonction "step" par phase (horloge de base)
    - Pas de "if"
    - Boucle "while" itère sur les "step" dans l'ordre
  - Une fonction "step" pour toute la période (horloge la + lente)
    - Transformation: Expansion d'hyper-période
    - Similaire à une transformation d'unrolling
    - Demande l'exposition de l'état interne

Spécification manuelle des phases

- Phases = schedule large-grain sur une période
  - → "Bon" choix de phase dépend de l'architecture
    Séquentiel: ≈ équilibrage de WCET
    Parallèle: . . . plus compliqué (≈ chemin critique)

Spécification manuelle des phases

- Phases = schedule large-grain sur une période
  - → "Bon" choix de phase dépend de l'architecture
    Séquentiel: ≈ équilibrage de WCET
    Parallèle: . . . plus compliqué (≈ chemin critique)
- Manipulation de phase est lourde à écrire et à modifier:
  - Une modification impacte plusieurs équations
  - Manuellement impossible pour grandes applis

Spécification manuelle des phases

- Phases = schedule large-grain sur une période
  - → "Bon" choix de phase dépend de l'architecture
    Séquentiel: ≈ équilibrage de WCET
    Parallèle: . . . plus compliqué (≈ chemin critique)
- Manipulation de phase est lourde à écrire et à modifier:
  - Une modification impacte plusieurs équations
  - Manuellement impossible pour grandes applis

⇒ Choix des phases devrait être séparé du calcul

Spécification manuelle des phases

- Phases = schedule large-grain sur une période
  - → "Bon" choix de phase dépend de l'architecture
    Séquentiel: ≈ équilibrage de WCET
    Parallèle: . . . plus compliqué (≈ chemin critique)
- Manipulation de phase est lourde à écrire et à modifier:
  - Une modification impacte plusieurs équations
  - Manuellement impossible pour grandes applis

⇒ Choix des phases devrait être séparé du calcul

- **Modification proposée:**
  - Possibilité de donne que la période des variables locales
  - Buffer implicite au niveau des équations
    (horloge rhs $<:$ horloge lhs)
- **Flot de compilation:**
  - Calcul d'horloge regroupe les contraintes sur les phases
  - Résolution des contraintes (étant donné une fonction de coût)
  - Utilise solution pour expliciter les phases et buffer ($\rightarrow$ delay)

Extraction des contraintes du calcul des horloges

- **buffer:** "delay" d'une longueur inconnue
  - $(0^k 10^{n-k-1}) <: (0^l 10^{m-l-1})$ iff $m = n$ et $k \leq l$

$$\frac{H \vdash a :: (0^k 10^{n-k-1}) \quad 0 \leq k \leq l < n}{H \vdash \text{buffer } a :: (0^l 10^{n-l-1})}$$

Extraction des contraintes du calcul des horloges

- **buffer:** "delay" d'une longueur inconnue
  - $(0^k 10^{n-k-1}) <: (0^l 10^{m-l-1})$ iff $m = n$ et $k \leq l$

$$\frac{H \vdash a :: (0^k 10^{n-k-1}) \quad 0 \leq k \leq l < n}{H \vdash \text{buffer } a :: (0^l 10^{n-l-1})}$$

- **bufferfby:** initialisation (période franchie)
- Variations du buffer avec d'autres contraintes:
  - Fixe la phase en sortie (ex: $p \leq 3$)
  - Contrainte sur la latence (ex: $p_B - p_A \leq 3$)

Extraction des contraintes du calcul des horloges

- **buffer:** "delay" d'une longueur inconnue
  - $(0^k 10^{n-k-1}) <: (0^l 10^{m-l-1})$ iff $m = n$ et $k \leq l$

$$\frac{H \vdash a :: (0^k 10^{n-k-1}) \quad 0 \leq k \leq l < n}{H \vdash \text{buffer } a :: (0^l 10^{n-l-1})}$$

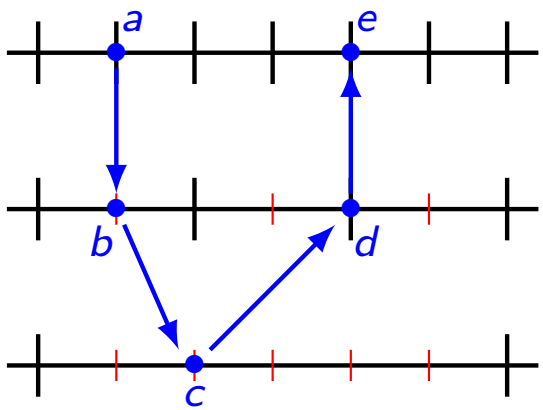
- **bufferfby:** initialisation (période franchie)
- Variations du buffer avec d'autres contraintes:
  - Fixe la phase en sortie (ex: $p \leq 3$)
  - Contrainte sur la latence (ex: $p_B - p_A \leq 3$)

- Au niveau des équations:
  - Règle originale: même horloges des 2 côtés de l'égalité

  $\rightarrow$ buffer nécessaire si phase de gauche inconnue

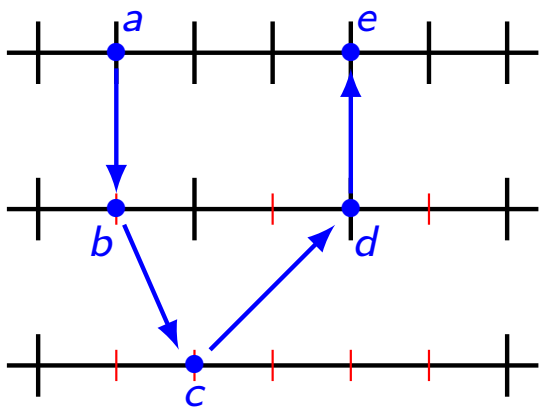
Exemple d'extraction de contrainte

$$
\begin{aligned}
\text{a,e} &:: \text{period}(1);\\
\text{b,d} &:: \text{period}(2);\\
\text{c} &:: \text{period}(6);\\
\text{b} &= \text{buffer } f_1(\text{a when (FT)});\\
\text{c} &= \text{buffer } f_2(\text{b when (TFF)});\\
\text{d} &= \text{buffer } f_3(\text{current((FFT), 0, c)});\\
\text{e} &= \text{buffer } f_4(\text{current((TF), 0, d)});
\end{aligned}
$$



Exemple d'extraction de contrainte

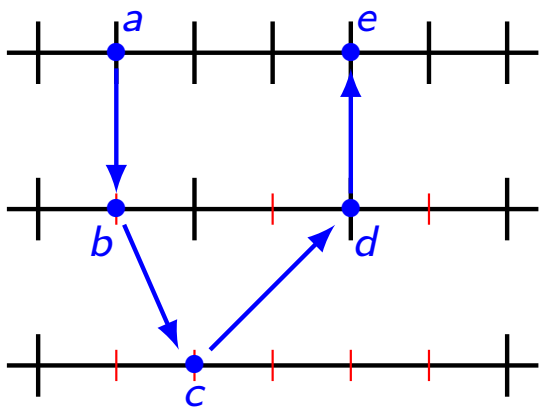
a,e :: period(1);
b,d :: period(2);
c :: period(6);
b = buffer f_1(a when (FT));
c = buffer f_2(b when (TFF));
d = buffer f_3(current((FFT), 0, c));
e = buffer f_4(current((TF), 0, d));



- Contraintes venant des déclarations de variables:
  $0 \leq p_a, p_e < 1$ / $0 \leq p_b, p_d < 2$ / $0 \leq p_c < 6$

Exemple d'extraction de contrainte

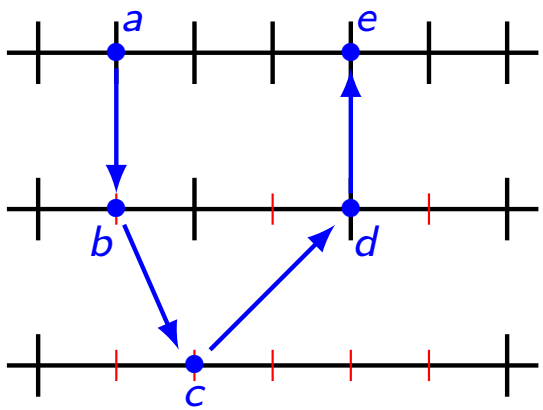
a,e :: period(1);
b,d :: period(2);
c :: period(6);
b = buffer f_1(a when (FT));
c = buffer f_2(b when (TFF));
d = buffer f_3(current((FFT), 0, c));
e = buffer f_4(current((TF), 0, d));



- Contraintes venant des déclarations de variables:
  $0 \leq p_a, p_e < 1$ / $0 \leq p_b, p_d < 2$ / $0 \leq p_c < 6$
- Contraintes venant des buffers:
  $p_a + 1 \leq p_b$ / $p_b \leq p_c$ / $p_c - 4 \leq p_d$ / $p_d \leq p_e$

Exemple d'extraction de contrainte

a,e :: period(1);
b,d :: period(2);
c :: period(6);
b = buffer f_1(a when (FT));
c = buffer f_2(b when (TFF));
d = buffer f_3(current((FFT), 0, c));
e = buffer f_4(current((TF), 0, d));



- Contraintes venant des déclarations de variables:
  $0 \leq p_a, p_e < 1$ / $0 \leq p_b, p_d < 2$ / $0 \leq p_c < 6$
- Contraintes venant des buffers:
  $p_a + 1 \leq p_b$ / $p_b \leq p_c$ / $p_c - 4 \leq p_d$ / $p_d \leq p_e$
- Ensemble des solutions:
  $p_a = p_e = 0$ / $p_b = 1$ / $p_d = 0$ / $1 \leq p_c \leq 4$

Résolution des contraintes (1)

- **Résolution:**
  - Forme des contraintes: permet une résolution efficace
  - Problème: fonction de coût peut les gâcher

Résolution des contraintes (1)

- **Résolution:**
  - Forme des contraintes: permet une résolution efficace
  - Problème: fonction de coût peut les gâcher

- **Cas d'étude:** application de commande de vol (6k nœuds, 30k données, 4 périodes harmoniques)
  - Fonction de coût pour le cas séquentiel: équilibrage (poids d'une tâche = son WCET)
  - Formulation de l'ILP possible mais tordue (astuces, dont une conversion d'entiers vers des booléens)

$\Rightarrow$ Ne passe pas à l'échelle...

Résolution des contraintes (2)

- Utiliser une ILP est overkill
  - Dans ce contexte, pas besoin de la solution optimale
  - Une solution "suffisamment bonne" suffit

Résolution des contraintes (2)

- Utiliser une ILP est overkill
  - Dans ce contexte, pas besoin de la solution optimale
  - Une solution "suffisamment bonne" suffit
- **Heuristique:**
  - Solution initiale: plus petites phases valides
  - Décroissance vers un minimum local:
    - Soft push (change une phase sans changer le reste)
    - Structure de donnée intermédiaire $\rightarrow$ évaluation rapide

Résolution des contraintes (2)

- Utiliser une ILP est overkill
  - Dans ce contexte, pas besoin de la solution optimale
  - Une solution "suffisamment bonne" suffit
- **Heuristique:**
  - Solution initiale: plus petites phases valides
  - Décroissance vers un minimum local:
    - Soft push (change une phase sans changer le reste)
    - Structure de donnée intermédiaire $\rightarrow$ évaluation rapide

$\Rightarrow$ Résultat: max sur une phase 0,6% au dessus moyenne
Temps pris par la décroissance quasi-instantanée

Résolution des contraintes (2)

- Utiliser une ILP est overkill
  - Dans ce contexte, pas besoin de la solution optimale
  - Une solution "suffisamment bonne" suffit
- **Heuristique:**
  - Solution initiale: plus petites phases valides
  - Décroissance vers un minimum local:
    - Soft push (change une phase sans changer le reste)
    - Structure de donnée intermédiaire $\rightarrow$ évaluation rapide

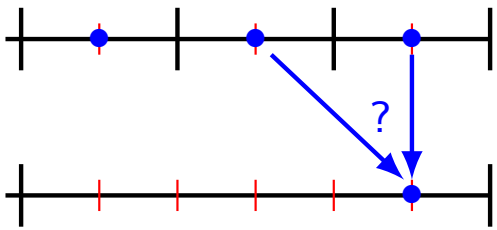
$\Rightarrow$ Résultat: max sur une phase 0,6% au dessus moyenne
Temps pris par la décroissance quasi-instantanée

- **Réinjection:**
  - Complète les horloges
  - Remplace les "buffer" par des "delay" (ou les supprime)

Calcul non-déterministe

- Valeurs physiques avec peu de variations
  - Ex: température extérieure
  - Veut une valeur récente, mais pas forcément la dernière

  $\Rightarrow$ Contrainte de dépendance peut être relaxée

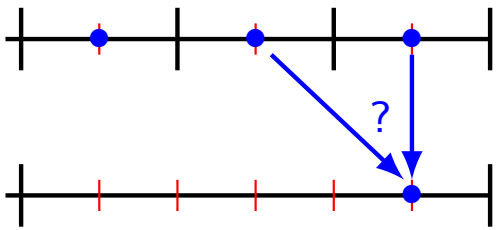


Contrainte voulue: $p_a + 2 \leq p_b$
(au lieu de $p_a + 4 \leq p_b$)

Calcul non-déterministe

- Valeurs physiques avec peu de variations
  - Ex: température extérieure
  - Veut une valeur récente, mais pas forcément la dernière

  $\Rightarrow$ Contrainte de dépendance peut être relaxée



Contrainte voulue: $p_a + 2 \leq p_b$
(au lieu de $p_a + 4 \leq p_b$)

- Comment l'exprimer de manière minimale dans le langage?

Opérateur non-déterministe: fby?

- **Proposition:** opérateur "fby?" pour contrôler le non-déterminisme
  - (Florence Maraninchi (Verimag) dans un contexte différent)

Opérateur non-déterministe: fby?

- **Proposition:** opérateur "fby?" pour contrôler le non-déterminisme
  - (Florence Maraninchi (Verimag) dans un contexte différent)
- Valeur de "i fby? expr" peut être:
  - soit "expr"
  - soit "i fby expr"

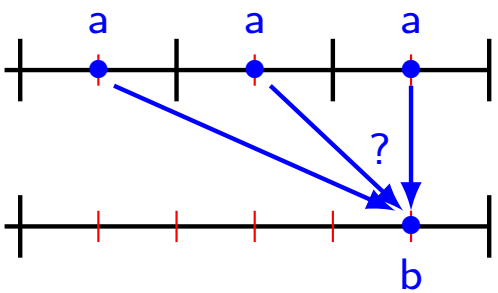
Opérateur non-déterministe: fby?

- **Proposition:** opérateur "fby?" pour contrôler le non-déterminisme
  - (Florence Maraninchi (Verimag) dans un contexte différent)
- Valeur de "i fby? expr" peut être:
  - soit "expr"
  - soit "i fby expr"
- Passage à la puissance: Valeur de "i fby?(n) expr" peut être:
  - soit "expr"
  - soit l'un des "i fby^k expr" (with $0 \leq k \leq n$)

Opérateur non-déterministe: fby?

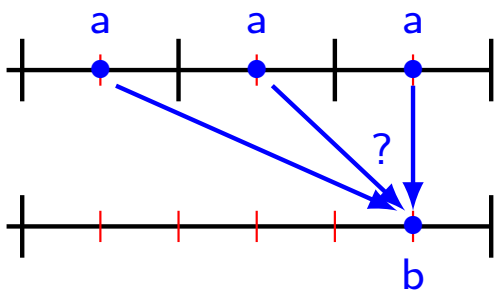
- **Proposition:** opérateur "fby?" pour contrôler le non-déterminisme
  - (Florence Maraninchi (Verimag) dans un contexte différent)
- Valeur de "i fby? expr" peut être:
  - soit "expr"
  - soit "i fby expr"
- Passage à la puissance: Valeur de "i fby?(n) expr" peut être:
  - soit "expr"
  - soit l'un des "i fby^k expr" (with $0 \leq k \leq n$)
- Déterminisme: remplacer les fby? par une valeur (dans notre cas: une fois les phases déterminées)

Extraction de contraintes avec du non-déterminisme



$$b = (0 \text{ fby}?^2 \text{ a}) \text{ when (FFT)};$$
$$p_a \leq p_b$$

Extraction de contraintes avec du non-déterminisme



b = (0 fby?^{2} a) when (FFT);

$p_a \leq p_b$

- Pour récupérer la bonne contrainte, doit regarder:
  - Le nombre de fby? au dessous d'un when
  - Le nombre de fby? au dessus d'un current
- **Résolution:**
  - Résolution comme précédemment
  - Déterminise en prenant la valeur la plus récente

En bref...

- Trois extensions incrémentales:
  - Horloges 1-synchrones
  - ...avec des phases inconnues
  - ...avec des calculs non-déterministes

En bref...

- Trois extensions incrémentales:
  - Horloges 1-synchrones
  - ...avec des phases inconnues
  - ...avec des calculs non-déterministes

- Travaux dans le futur proche:
  - Autre grand UC (de taille réelle) à faire tourner
  - Fonction de coût: minimisation du chemin critique par phase
    - Possible de ruser avec le même genre d'heuristique?

En bref...

- Trois extensions incrémentales:
  - Horloges 1-synchrones
  - ...avec des phases inconnues
  - ...avec des calculs non-déterministes
- Travaux dans le futur proche:
  - Autre grand UC (de taille réelle) à faire tourner
  - Fonction de coût: minimisation du chemin critique par phase
    - Possible de ruser avec le même genre d'heuristique?
- Merci de votre attention ...
  - Avez-vous des questions?