

# Ré-identification sans coordination dans les types de données répliquées sans conflits

---

Matthieu Nicolas ([matthieu.nicolas@loria.fr](mailto:matthieu.nicolas@loria.fr))

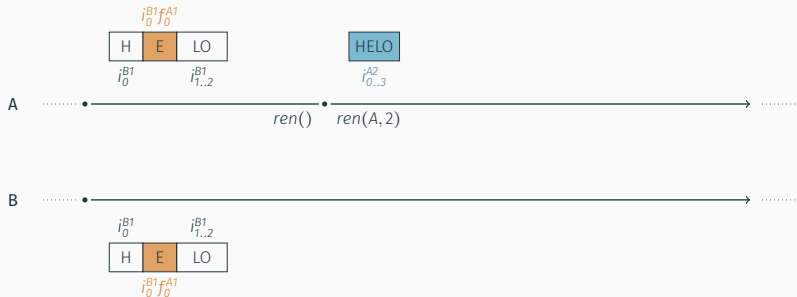
20 décembre 2022

<i>Rapporteurs :</i>	Hanifa Boucheneb Davide Frey	Professeure, Polytechnique Montréal Chargé de recherche, HdR, Inria Rennes Bretagne-Atlantique
<i>Examineurs :</i>	Hala Skaf-Molli Stephan Merz	Maîtresse de conférences, HdR, Nantes Université, LS2N Directeur de Recherche, Inria Nancy - Grand Est
<i>Encadrants :</i>	Olivier Perrin Gérald Oster	Professeur des Universités, Université de Lorraine, LORIA Maître de conférences, Université de Lorraine, LORIA

# RenamableLogootSplit

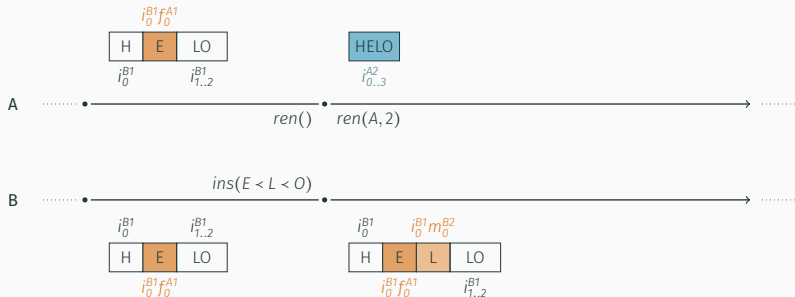
---

# Intégration des opérations *insert* et *remove* concurrentes



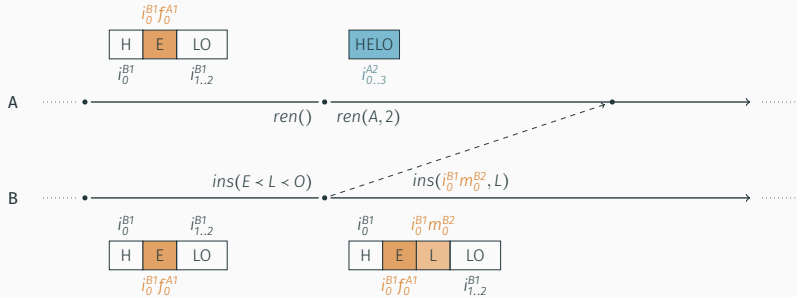
- Peuvent générer opérations concurrentes aux opérations *rename*

# Intégration des opérations *insert* et *remove* concurrentes



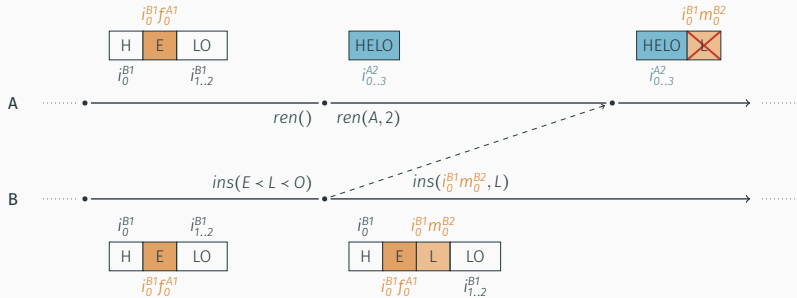
- Peuvent générer opérations concurrentes aux opérations *rename*

# Intégration des opérations *insert* et *remove* concurrentes



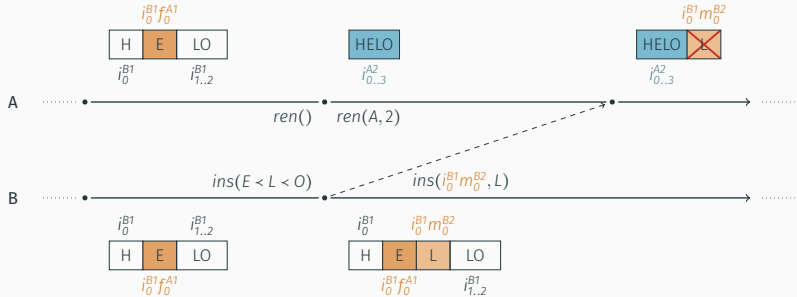
- Peuvent générer opérations concurrentes aux opérations *rename*

# Intégration des opérations *insert* et *remove* concurrentes



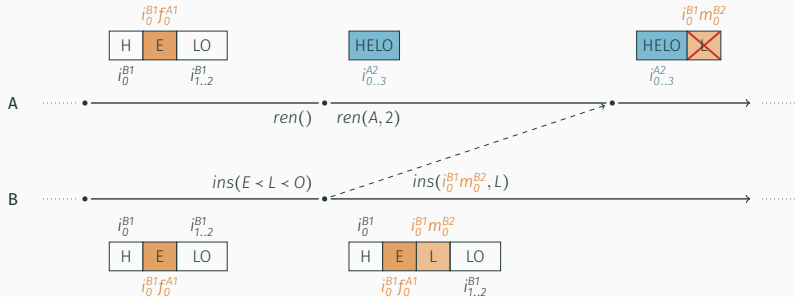
- Peuvent générer opérations concurrentes aux opérations *rename*

# Intégration des opérations *insert* et *remove* concurrentes



- Peuvent générer opérations concurrentes aux opérations *rename*
- Produisent anomalies si intégrées naïvement

# Intégration des opérations *insert* et *remove* concurrentes

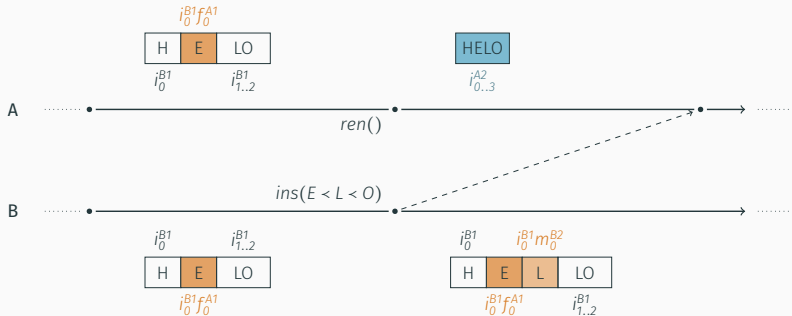


- Peuvent générer opérations concurrentes aux opérations *rename*
- Produisent anomalies si intégrées naïvement

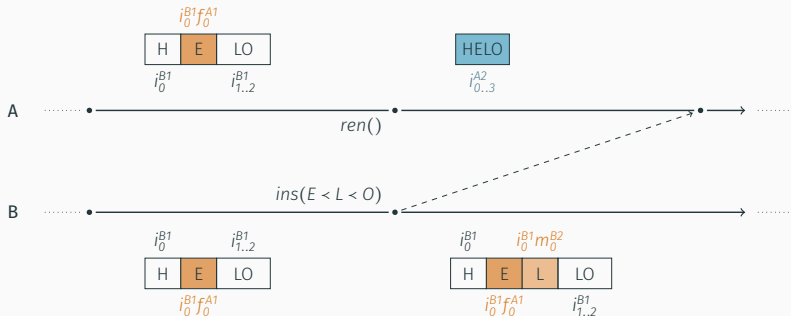
Nécessité d'un mécanisme dédié



# Détection des opérations concurrentes à opération *rename*

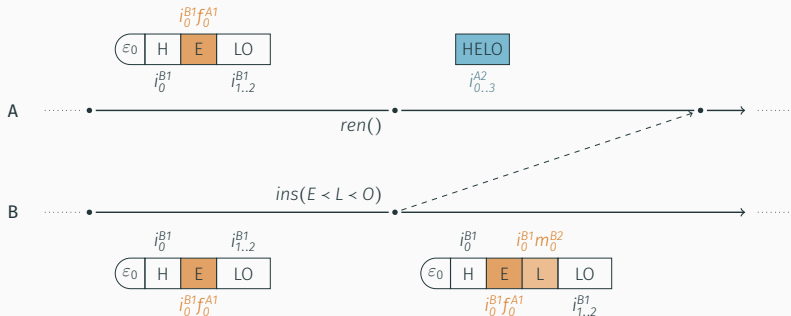


# Détection des opérations concurrentes à opération *rename*



Ajout mécanisme d'époques

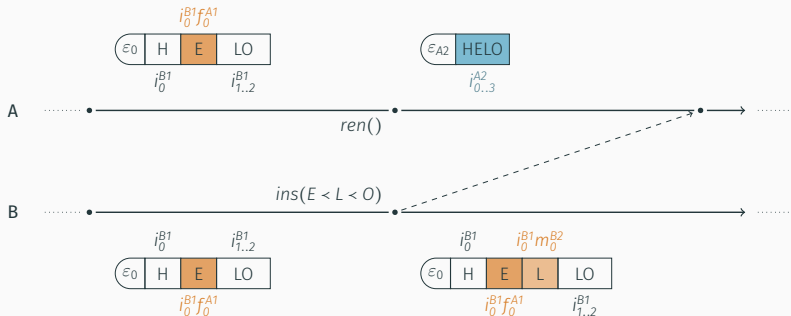
# Détection des opérations concurrentes à opération *rename*



## Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée  $\epsilon_0$

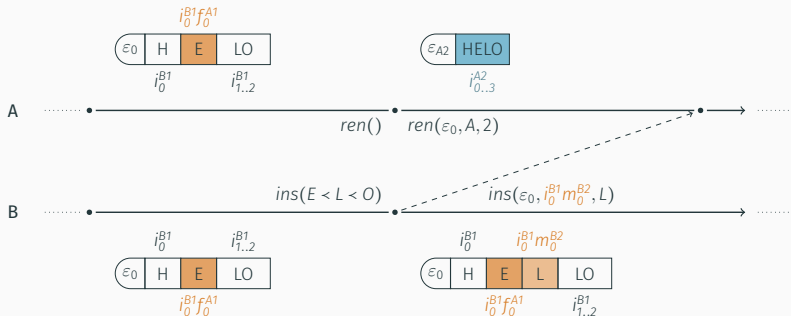
# Détection des opérations concurrentes à opération *rename*



## Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée  $\epsilon_0$
- *rename* font progresser à nouvelle époque,  $\epsilon_{nodeId \ nodeSeq}$

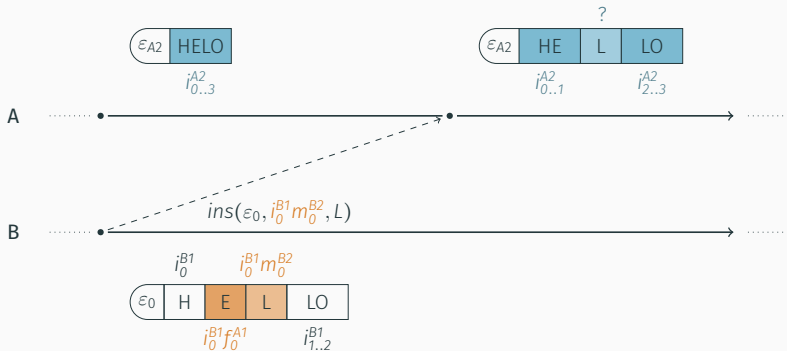
# Détection des opérations concurrentes à opération *rename*



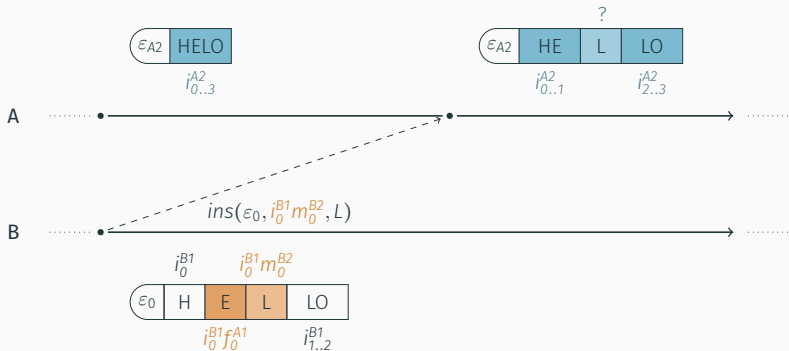
## Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée  $\epsilon_0$
- *rename* font progresser à nouvelle époque,  $\epsilon_{nodeId \ nodeSeq}$
- Opérations labellisées avec époque de génération

# Gestion des opérations *insert* et *remove* concurrentes

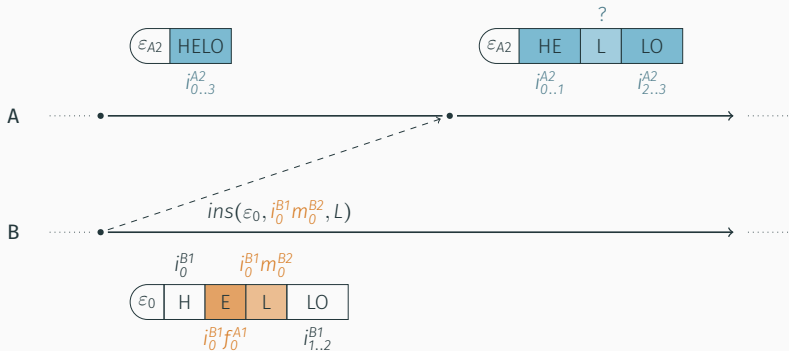


# Gestion des opérations *insert* et *remove* concurrentes



Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

# Gestion des opérations *insert* et *remove* concurrentes

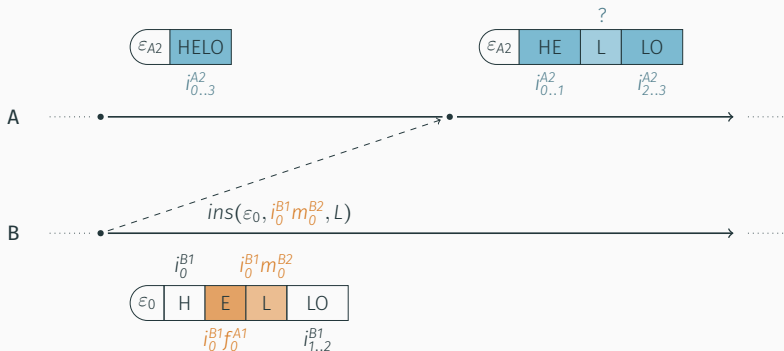


Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

- Prend la forme de l'algorithme `renameId`



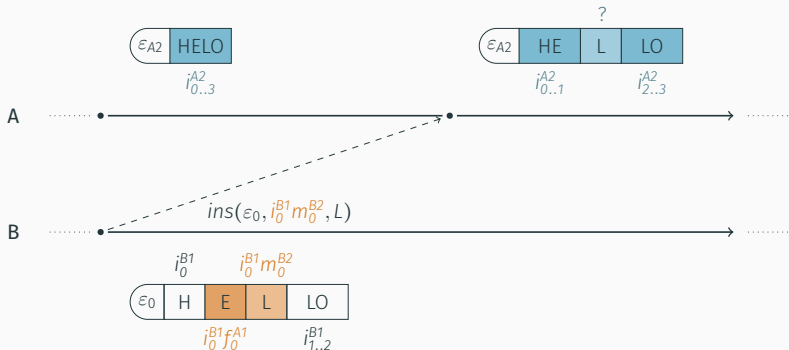
# Gestion des opérations *insert* et *remove* concurrentes



Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

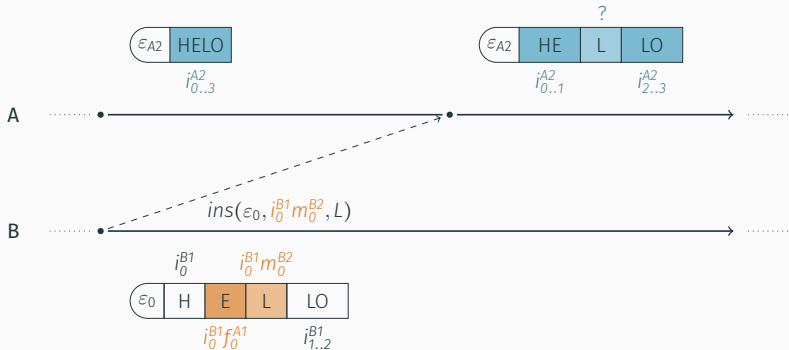
- Prend la forme de l'algorithme `renameId`
- Inclure l'effet de l'opération *rename* dans l'opération transformée

# Exemple de renameId



Exemple avec  $i_0^{B1}m_0^{B2}$

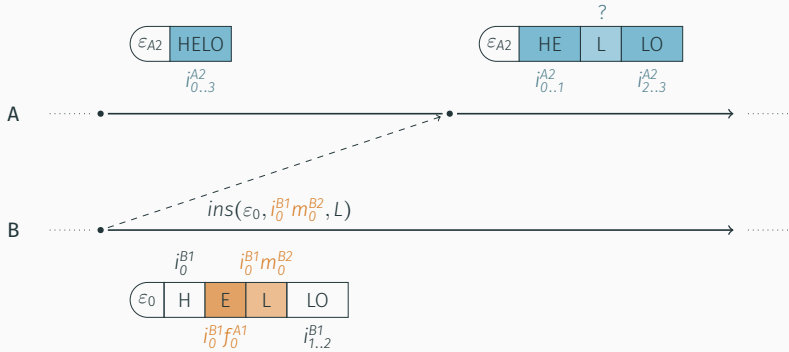
# Exemple de renameId



## Exemple avec $i_0^{B1} m_0^{B2}$

- Trouver son prédécesseur à l'époque d'origine  $\epsilon_0$  :  $i_0^{B1} f_0^{A1}$

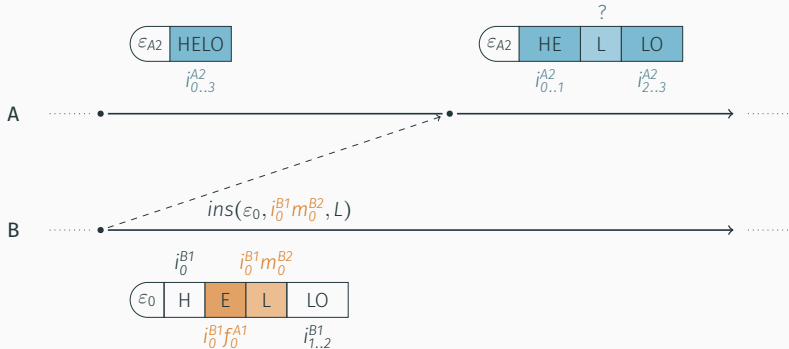
# Exemple de renameId



## Exemple avec $i_0^{B1} m_0^{B2}$

- Trouver son prédécesseur à l'époque d'origine  $\varepsilon_0$  :  $i_0^{B1} f_0^{A1}$
- Trouver son équivalent à l'époque cible  $\varepsilon_{A2}$  :  $i_1^{A2}$

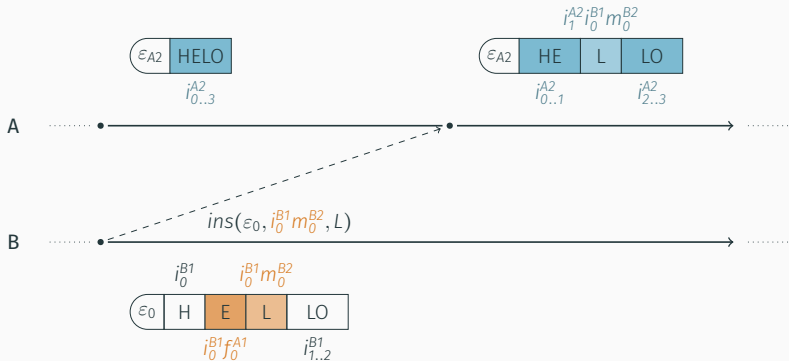
# Exemple de renameId



## Exemple avec $i_0^{B1} m_0^{B2}$

- Trouver son prédécesseur à l'époque d'origine  $\epsilon_0$  :  $i_0^{B1} f_0^{A1}$
- Trouver son équivalent à l'époque cible  $\epsilon_{A2}$  :  $i_1^{A2}$
- Concaténer ce dernier à  $i_0^{B1} m_0^{B2}$  pour obtenir son équivalent à  $\epsilon_{A2}$  :  $i_1^{A2} i_0^{B1} m_0^{B2}$

# Exemple de renameId

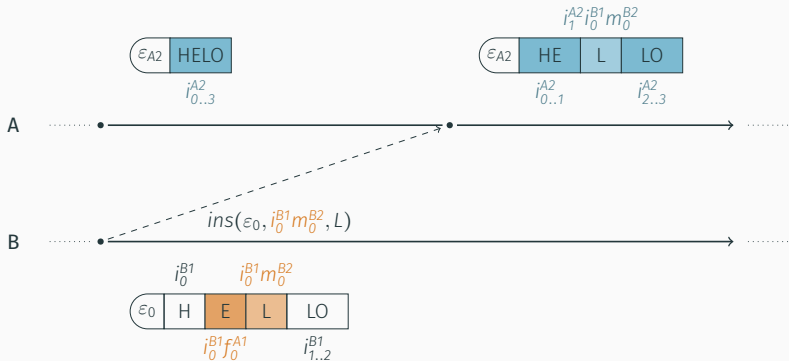


## Exemple avec $i_0^{B1} m_0^{B2}$

- Trouver son prédécesseur à l'époque d'origine  $\epsilon_0$  :  $i_0^{B1} f_0^{A1}$
- Trouver son équivalent à l'époque cible  $\epsilon_{A2}$  :  $i_1^{A2}$
- Concaténer ce dernier à  $i_0^{B1} m_0^{B2}$  pour obtenir son équivalent à  $\epsilon_{A2}$  :  $i_1^{A2} i_0^{B1} m_0^{B2}$

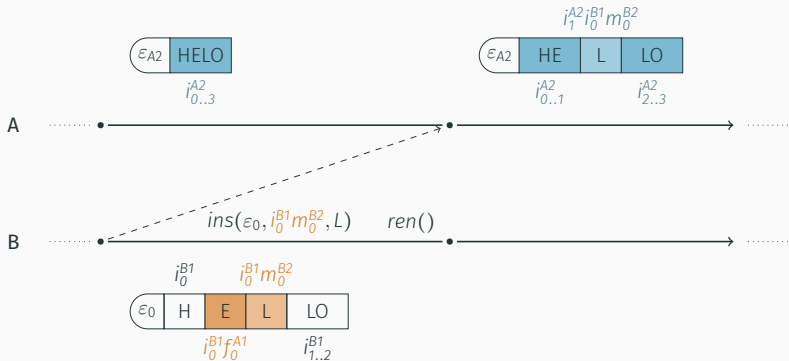
Et en cas d'opérations *rename* concurrentes?

# Opérations *rename* concurrentes

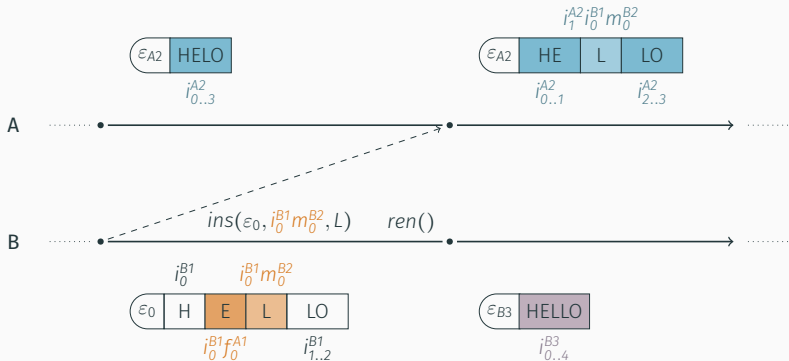




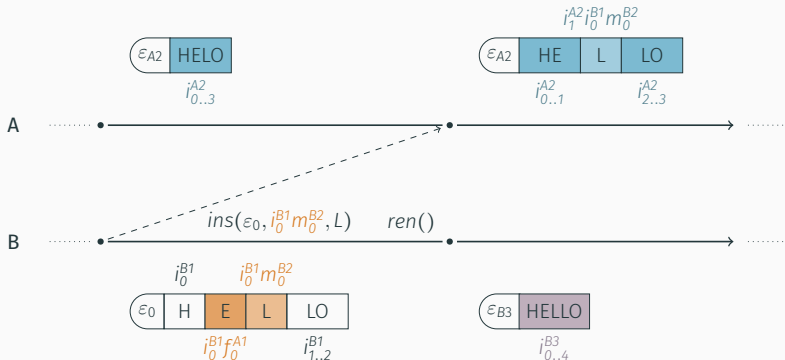
# Opérations *rename* concurrentes



# Opérations *rename* concurrentes

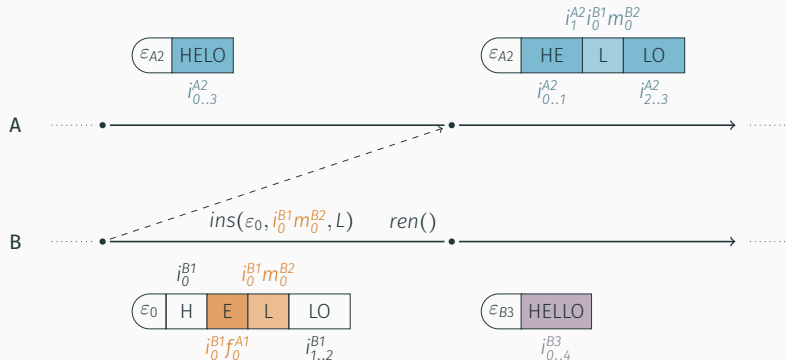


# Opérations *rename* concurrentes



Comment faire converger les noeuds ?

# Opérations *rename* concurrentes



Comment faire converger les noeuds ?

Besoin d'un mécanisme additionnel de résolution de conflits

## Observation

- Opérations *rename* sont des opérations systèmes...
- ...pas des opérations utilisateur-rices

# Résolution de conflits entre opérations *rename* concurrentes

## Observation

- Opérations *rename* sont des opérations systèmes...
- ...pas des opérations utilisateur-rices

## Proposition

- Considérer une opération *rename* comme prioritaire...
- ...et ignorer les opérations *rename* en conflit avec elle

# Algorithme d'intégration d'une opération *rename*

Intuition

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues



# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible
  - Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible
  - Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
  - Annuler l'effet des opérations *rename* de l'époque courante au PPAC

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible
  - Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
  - Annuler l'effet des opérations *rename* de l'époque courante au PPAC
  - Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

# Algorithme d'intégration d'une opération *rename*

## Intuition

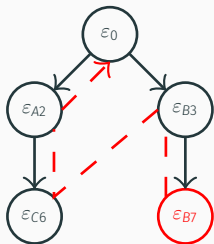
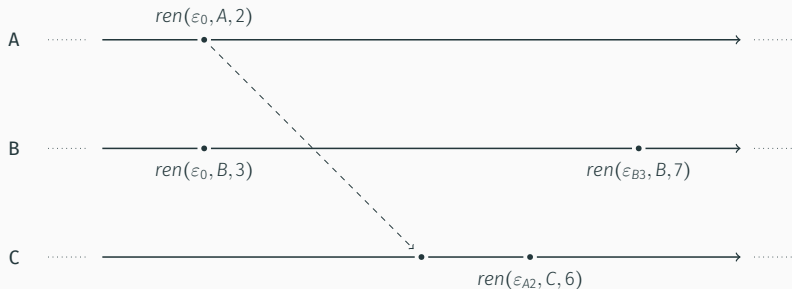
- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible
  - Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
  - Annuler l'effet des opérations *rename* de l'époque courante au PPAC
  - Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

# Algorithme d'intégration d'une opération *rename*

## Intuition

- Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
- Choisir entre époque courante et nouvelle époque **l'époque cible**
- Si changement d'époque cible
  - Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
  - Annuler l'effet des opérations *rename* de l'époque courante au PPAC
  - Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

# Choisir une époque comme époque cible



- Doit définir relation *priority*, notée  $<_{\epsilon}$ , ordre strict total sur les époques
- Utilise ordre lexicographique sur chemins des époques dans l'arbre



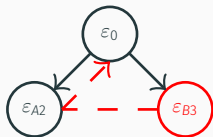
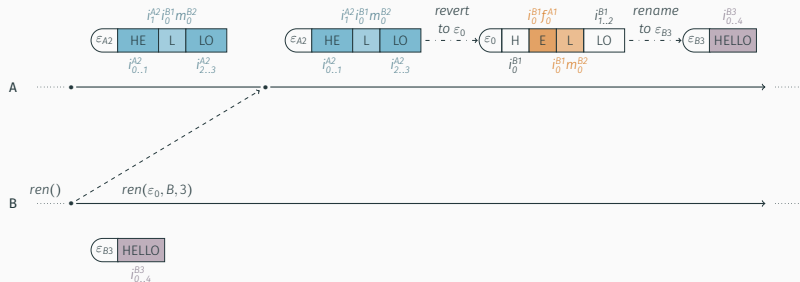
# Annuler l'effet d'une opération *rename*

- Propose un nouvel algorithme, *revertRenameId*
- Permet d'obtenir l'identifiant correspondant à l'époque parente

## Intuition

1. *id* fait partie des identifiants renommés : doit retourner son ancienne valeur
2. *id* a (potentiellement) été inséré en concurrence : doit restaurer sa (potentielle) ancienne valeur
3. *id* a été inséré après le renommage : doit retourner une valeur qui préserve l'ordre

# Opérations *rename* concurrentes



- TODO : Illustrer choix de l'époque cible, cas 1 et cas 2