Entretien technique

Matthieu Nicolas (nicolasmatthieu57@gmail.com)

19 octobre 2023

Déroulement de carrière

Ingénieur Recherche & Développement *2014-2017*

Inria, Loria, équipe Coast, Nancy

- MUTE : conception et développement d'un éditeur de texte collaboratif temps réel web pair-à-pair (`https://mute.loria.fr`)
- PLM : webification d'un environnement d'apprentissage de la programmation (`http://people.irisa.fr/Martin.Quinson/Teaching/PLM/`)

Doctorat en informatique *2017-2022*

Université de Lorraine, Loria, équipe Coast, Nancy

- (Ré)Identification efficace dans les types de données répliquées sans conflit (CRDTs)

Déroulement de carrière

Ingénieur Recherche & Développement *2014-2017*

Inria, Loria, équipe Coast, Nancy

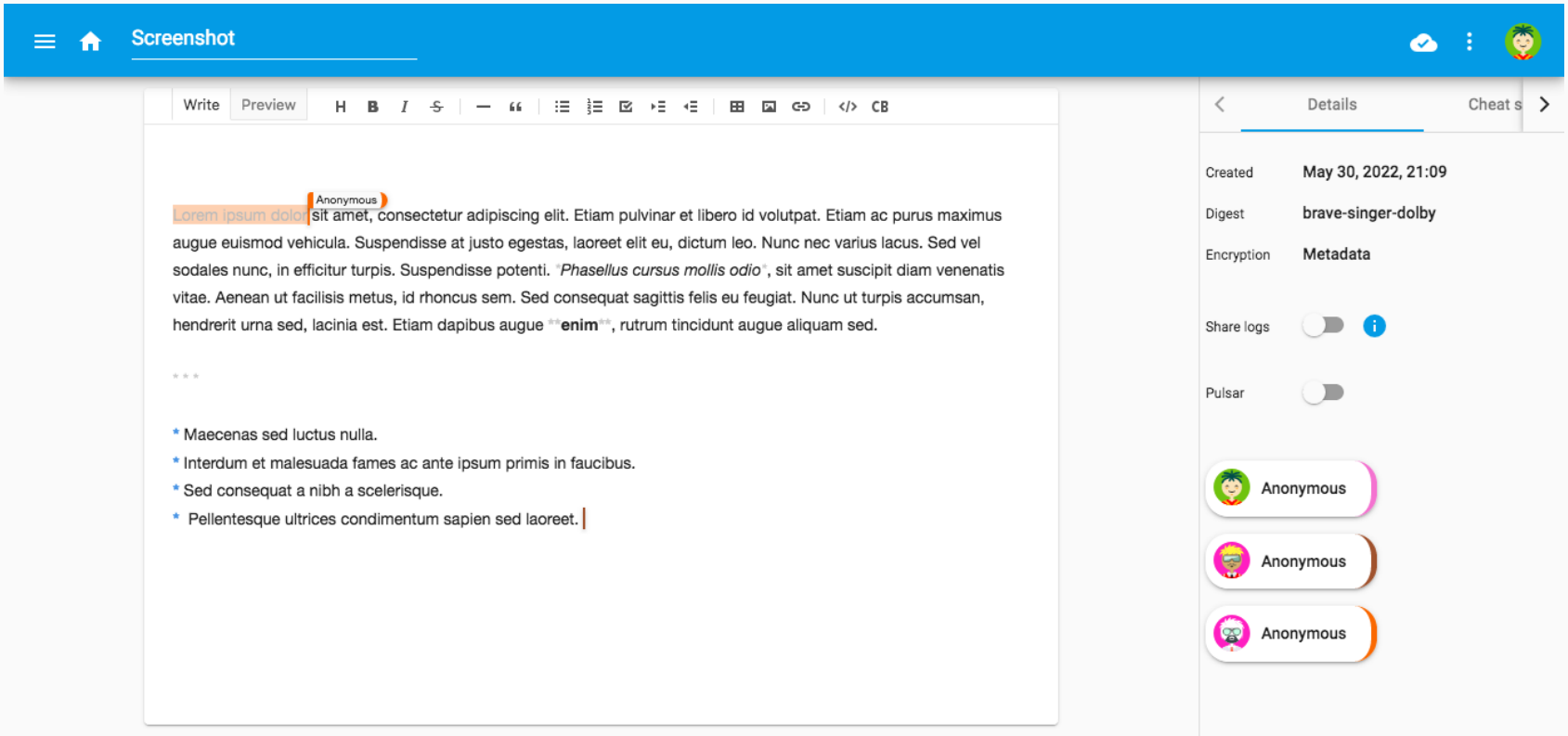
- MUTE : conception et développement d'un éditeur de texte collaboratif temps réel web pair-à-pair (`https://mute.loria.fr`)
- PLM : webification d'un environnement d'apprentissage de la programmation (`http://people.irisa.fr/Martin.Quinson/Teaching/PLM/`)

Doctorat en informatique *2017-2022*

Université de Lorraine, Loria, équipe Coast, Nancy

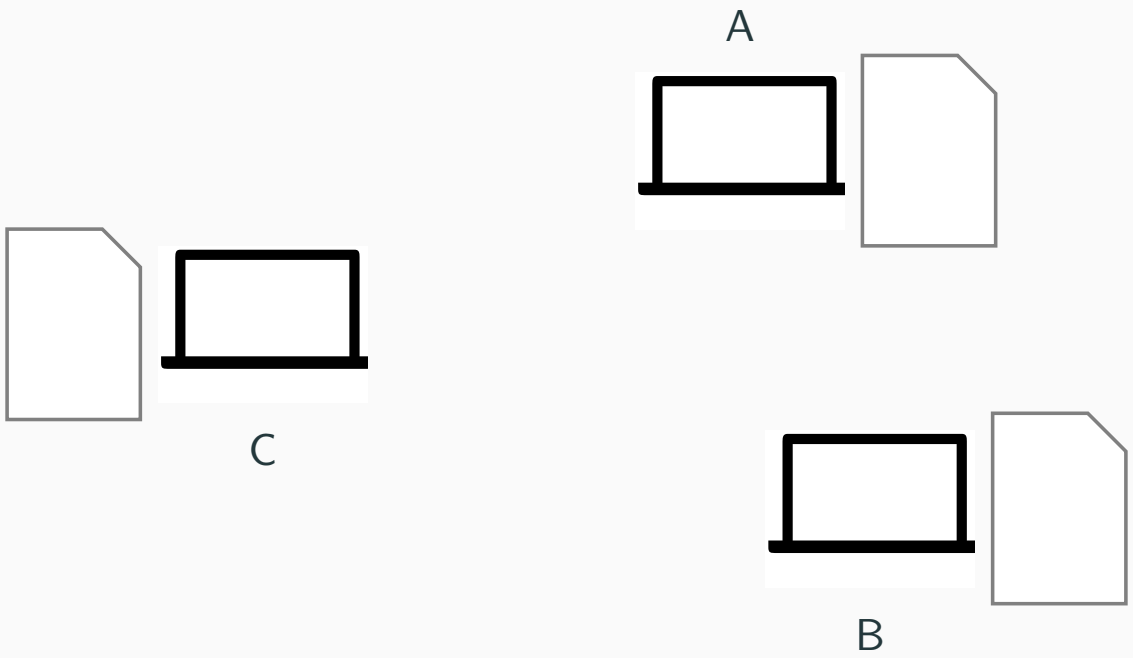
- (Ré)Identification efficace dans les types de données répliquées sans conflit (CRDTs)

MUTE, un exemple de Local-First Software (LFS) [Kle+19]

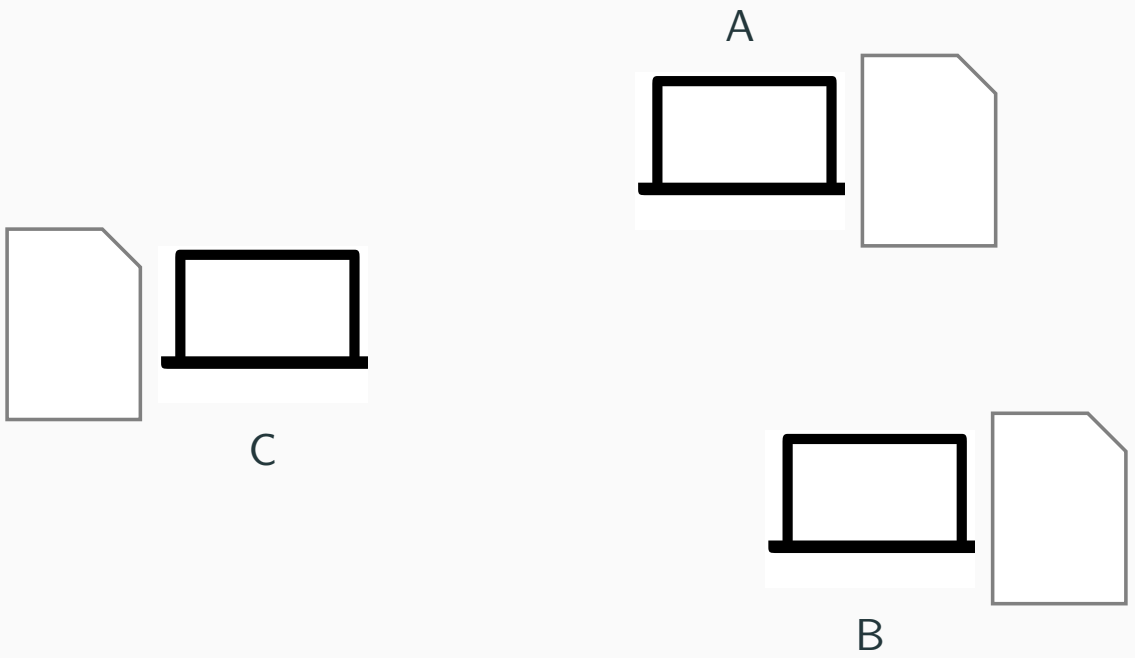


- Application pair-à-pair
- Permet de rédiger collaborativement des documents texte
- Garantit la confidentialité & souveraineté des données

Réplication dans applications collaboratives pair-à-pair

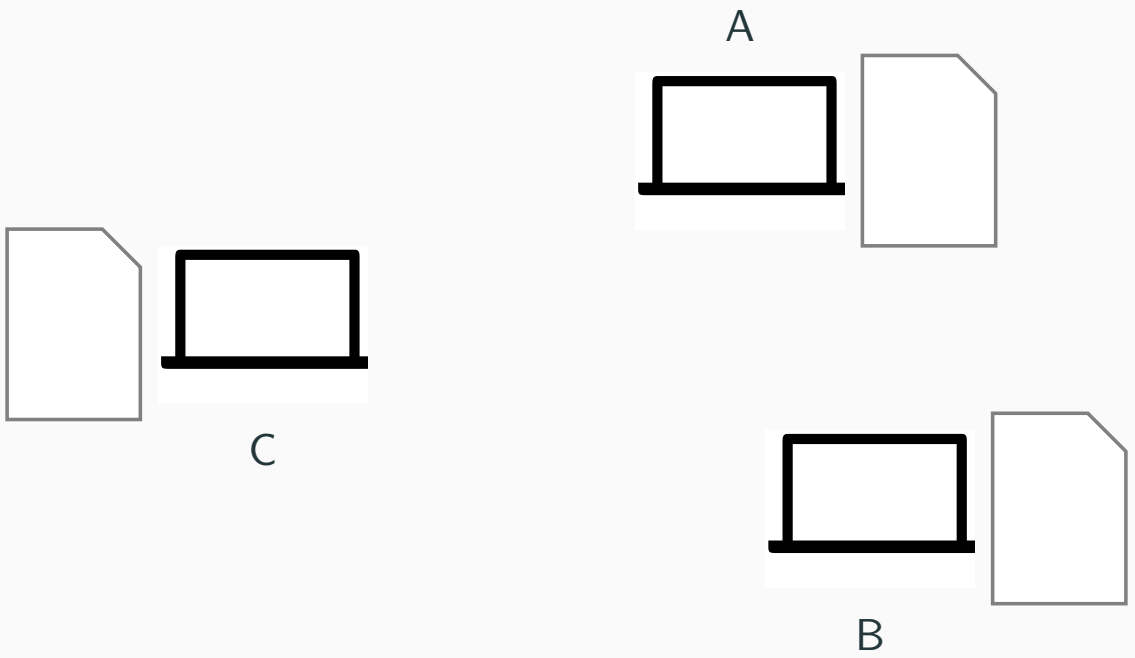


Réplication dans applications collaboratives pair-à-pair



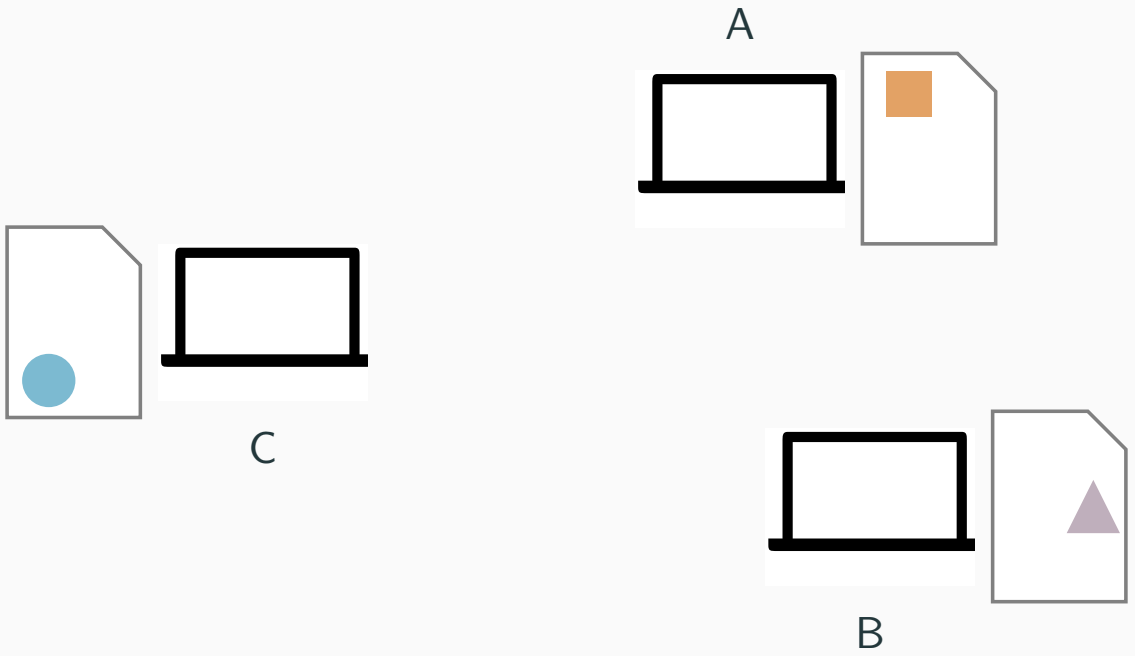
- Noeuds peuvent être déconnectés

Réplication dans applications collaboratives pair-à-pair



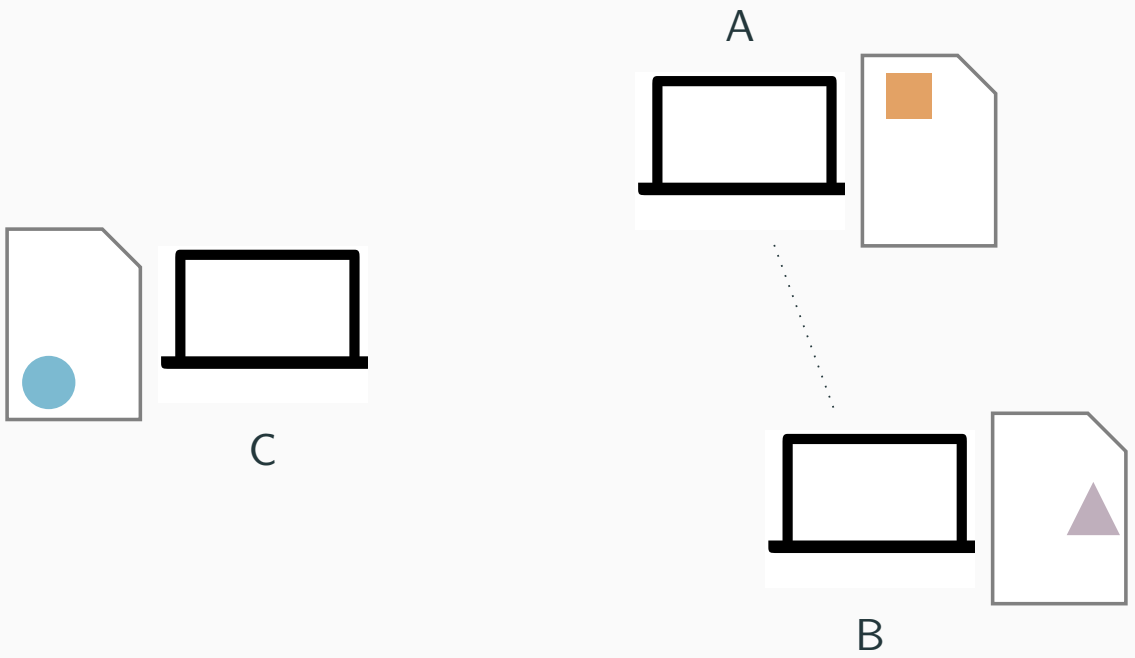
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



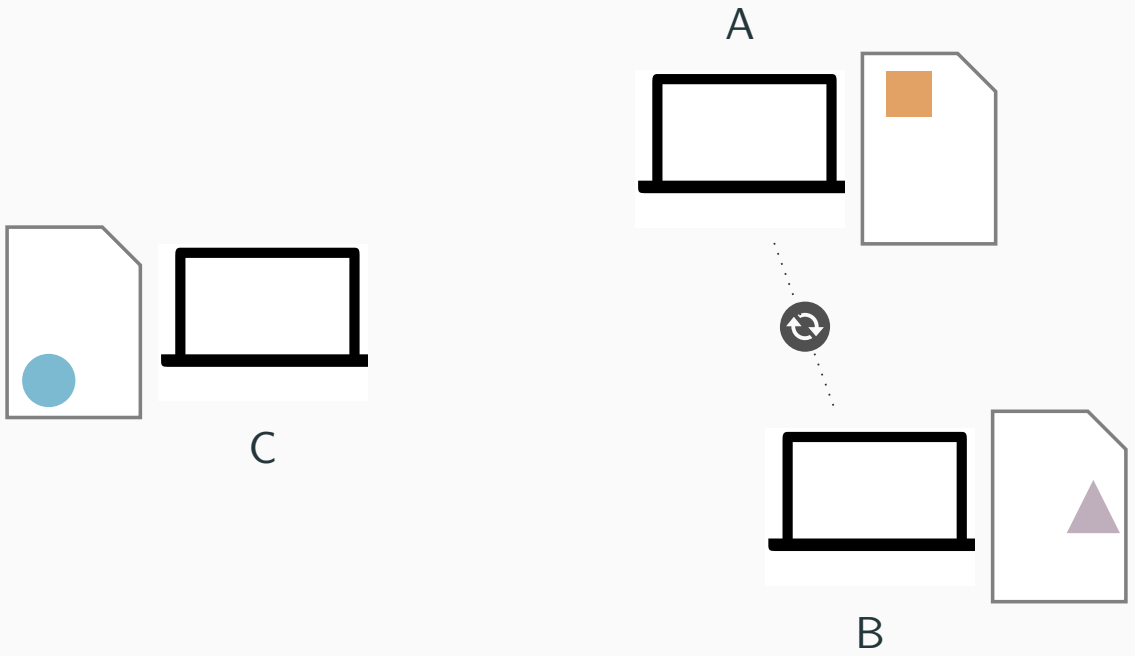
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



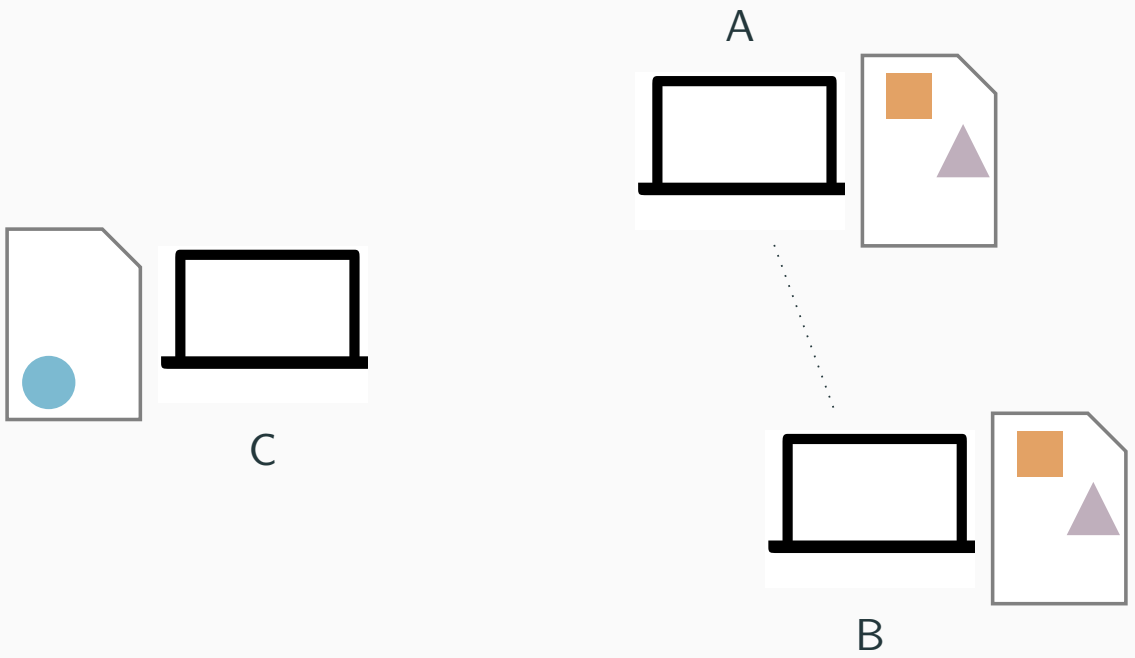
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



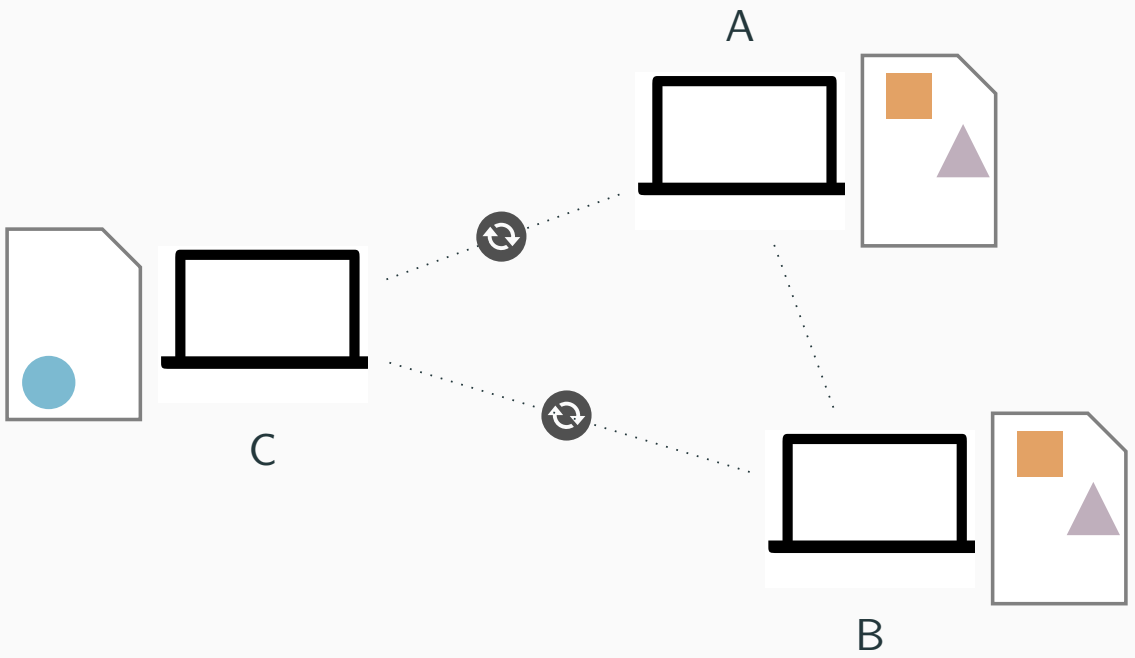
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



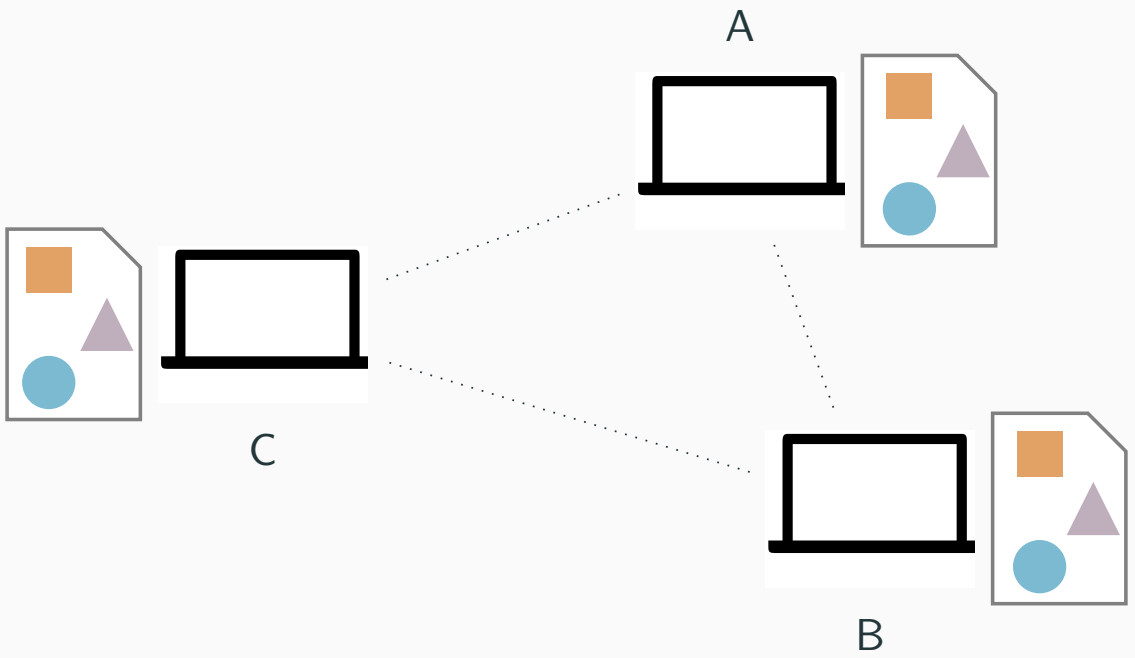
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



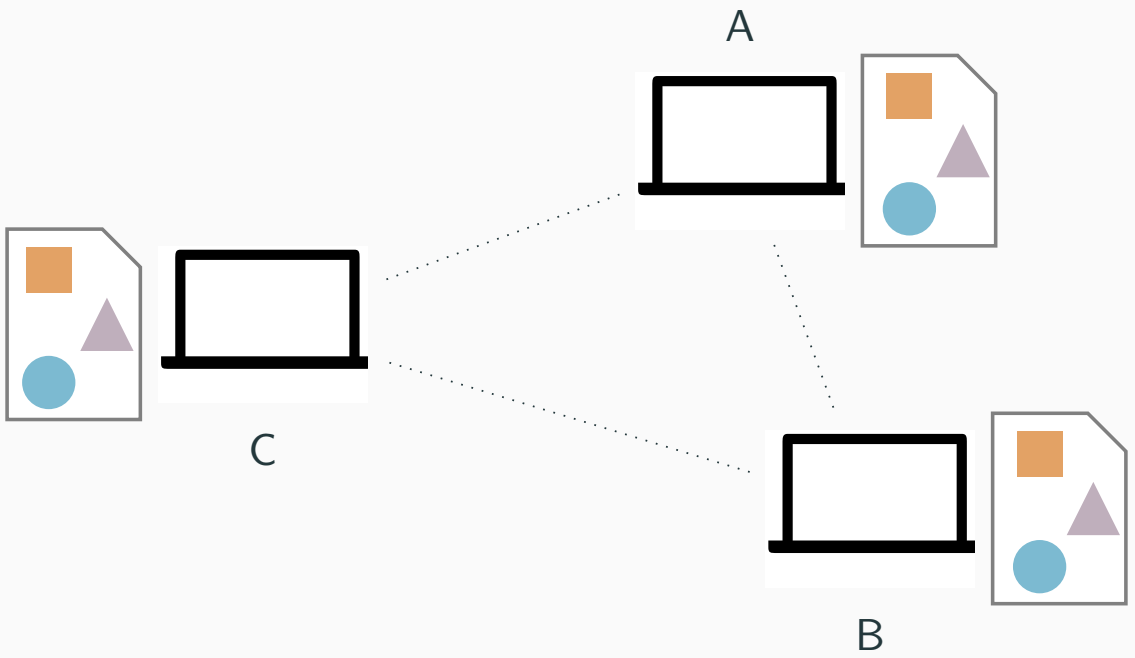
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)
- Doit garantir cohérence à terme [Ter+95]...
- ... malgré ordres différents d'intégration des modifications

Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)
- Doit garantir cohérence à terme [Ter+95]...
- ... malgré ordres différents d'intégration des modifications

Nécessite des *mécanismes de résolution de conflits*

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

Propriétés des CRDTs

- Permettent modifications sans coordination
- Garantissent la cohérence forte à terme

Conflict-free Replicated Data Types (CRDTs) [Sha+11]

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

Propriétés des CRDTs

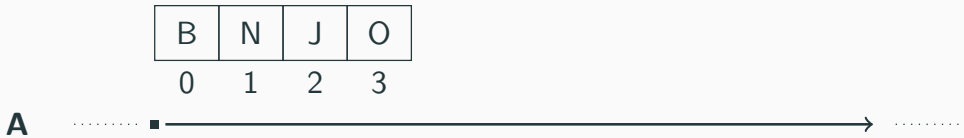
- Permettent modifications sans coordination
- Garantissent la cohérence forte à terme

Cohérence forte à terme

Ensemble des noeuds ayant intégrés le même ensemble de modifications obtient des états équivalents, sans nécessiter d'actions ou messages supplémentaires

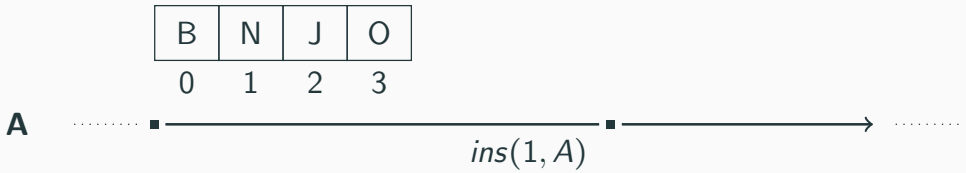
CRDTs pour le type Séquence

Type Séquence usuel



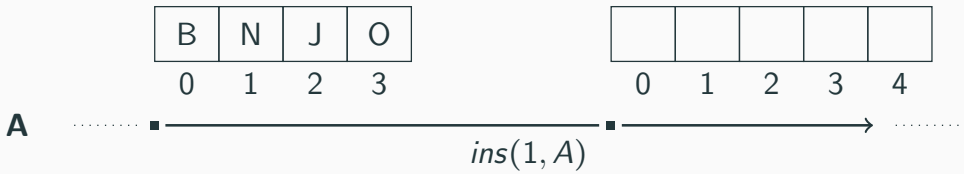
CRDTs pour le type Séquence

Type Séquence usuel



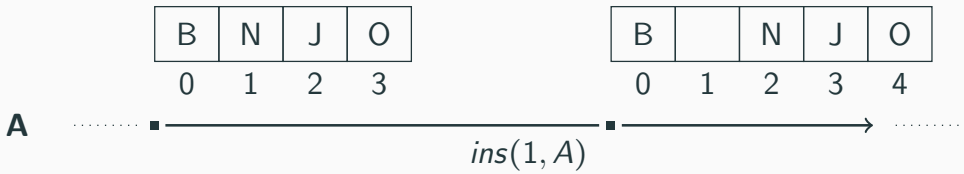
CRDTs pour le type Séquence

Type Séquence usuel



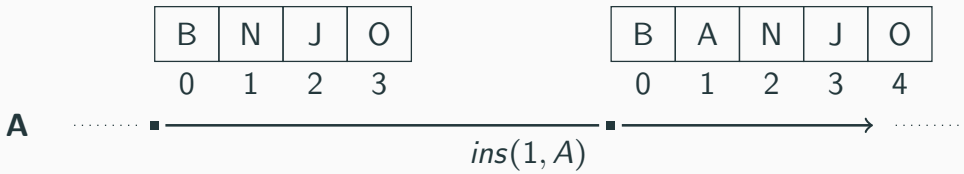
CRDTs pour le type Séquence

Type Séquence usuel



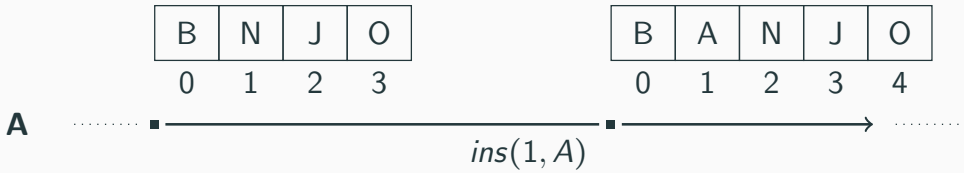
CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour le type Séquence

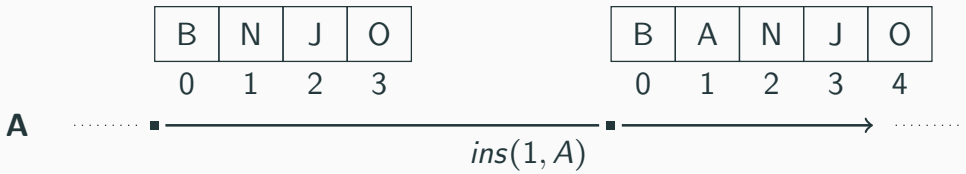
Type Séquence usuel



- Changements d'indices sont source de conflits

CRDTs pour le type Séquence

Type Séquence usuel

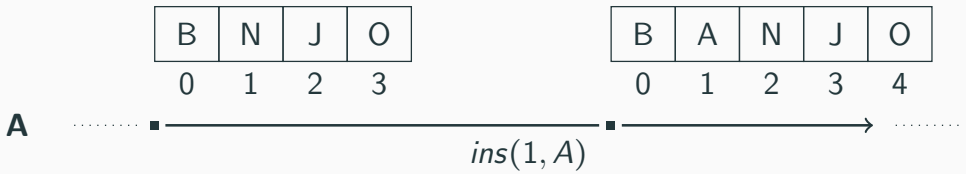


CRDTs pour Séquence

- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

CRDTs pour le type Séquence

Type Séquence usuel



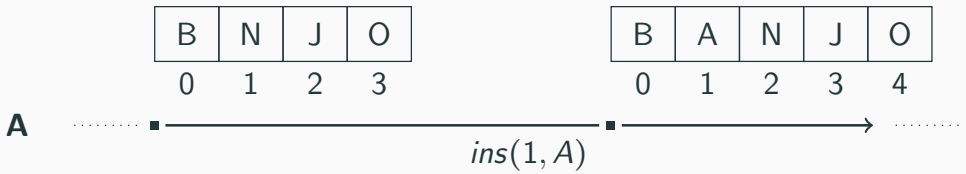
CRDTs pour Séquence

- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

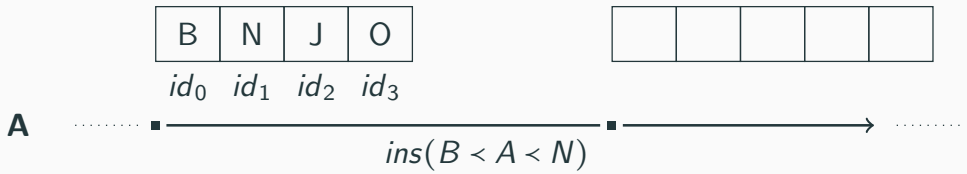
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence

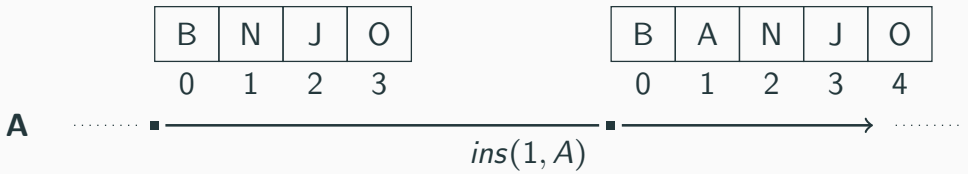


- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

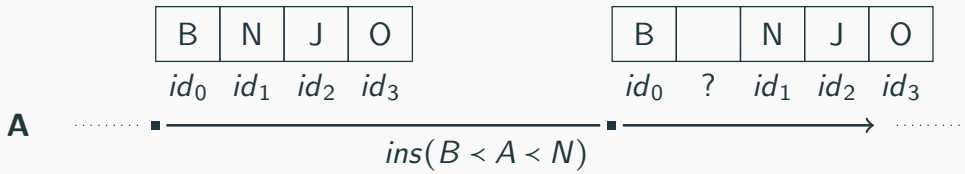
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



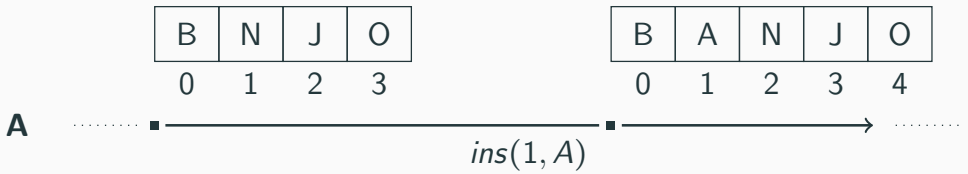
- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

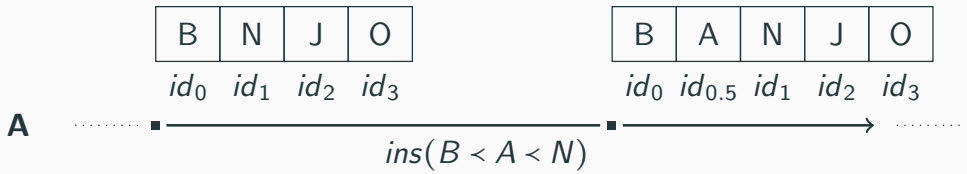
- Identifiants appartiennent à un espace dense

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

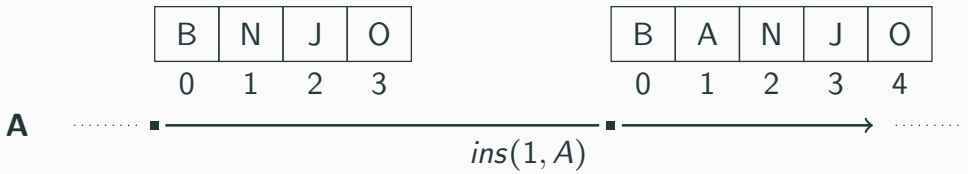
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Identifiants appartiennent à un espace dense

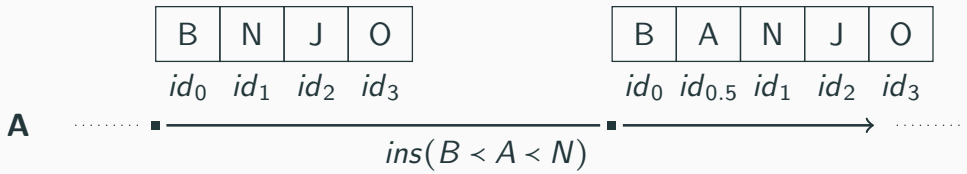
$$id_0 <_{id} id_{0.5} <_{id} id_1$$

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



- Changements d'indices sont source de conflits
- CRDTs assignent des identifiants de position [Pre+09] à chaque élément
- Identifiants permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Identifiants appartiennent à un espace dense

$$id_0 <_{id} id_{0.5} <_{id} id_1$$

Utilise LogootSplit [And+13] comme base

Identifiant LogootSplit

Identifiant

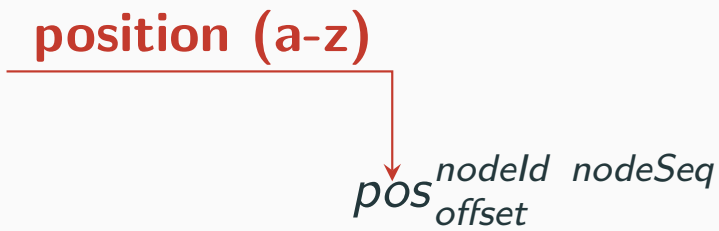
- Composé d'un ou plusieurs tuples de la forme

$$pos_{offset}^{nodeId\ \ nodeSeq}$$

Identifiant LogootSplit

Identifiant

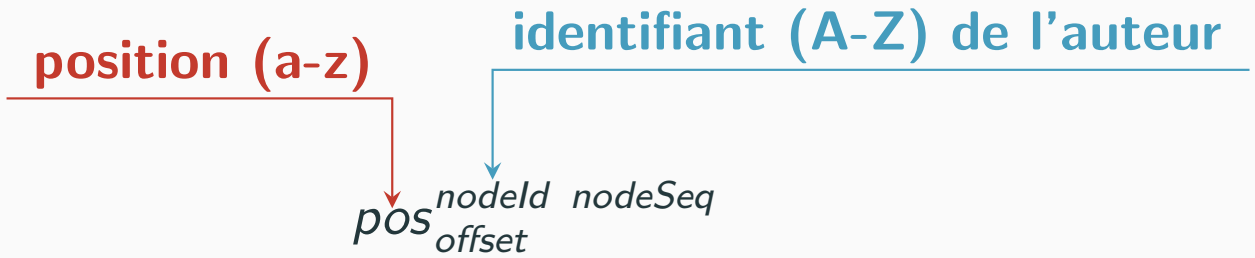
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

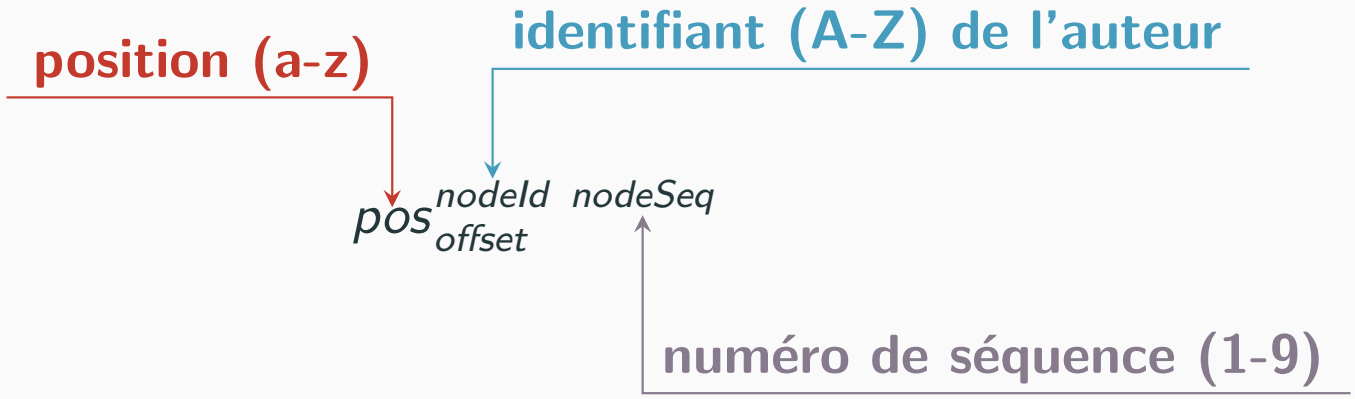
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

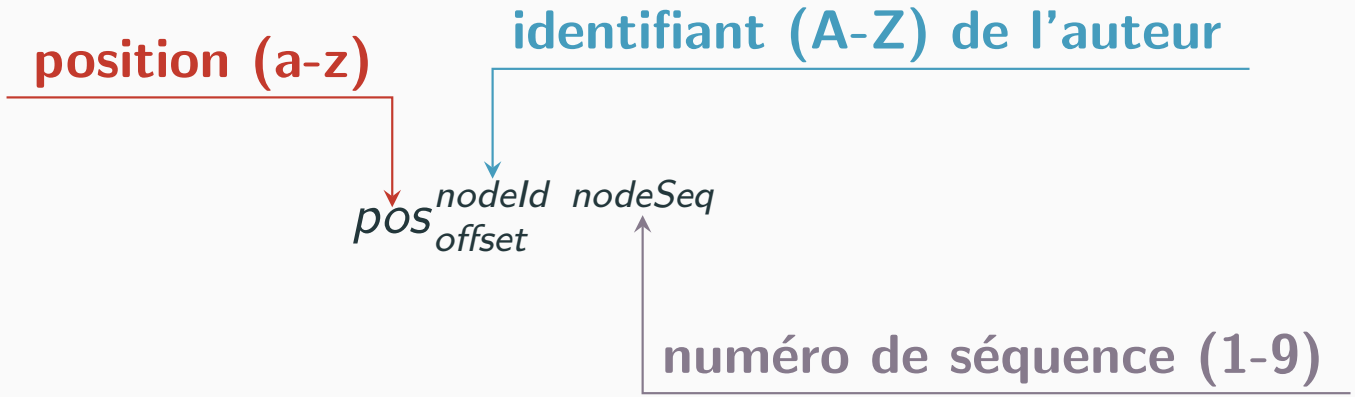
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



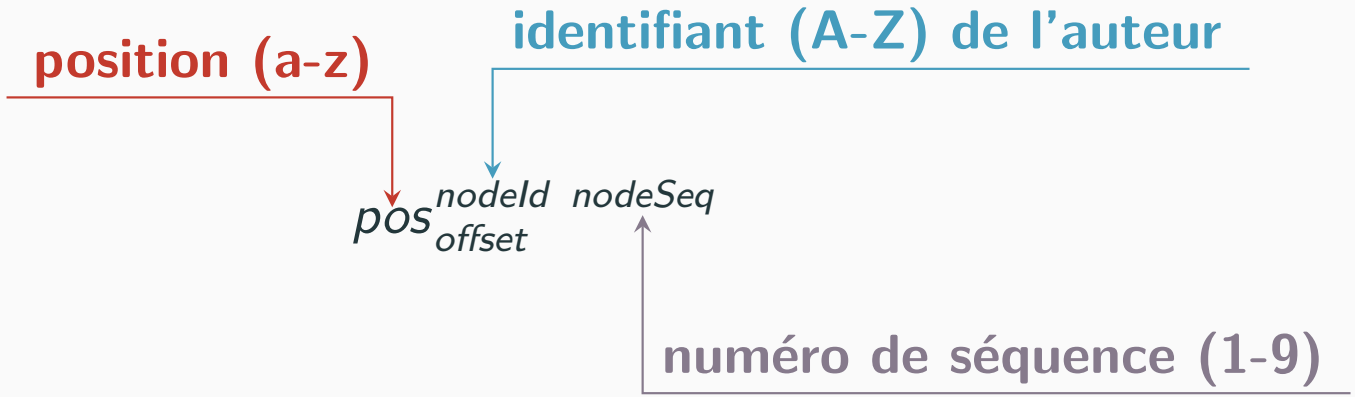
Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

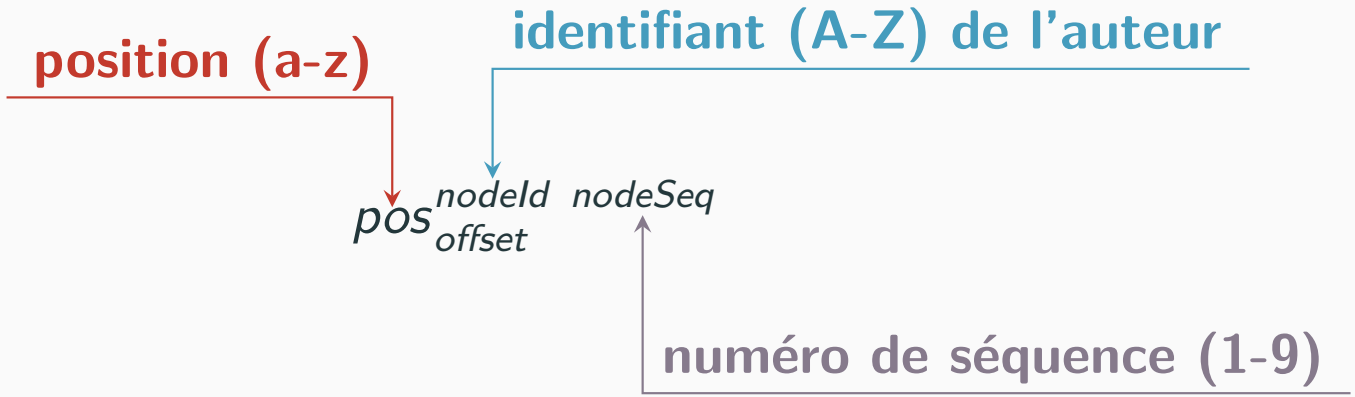
Exemples

$$d_0^{F5}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

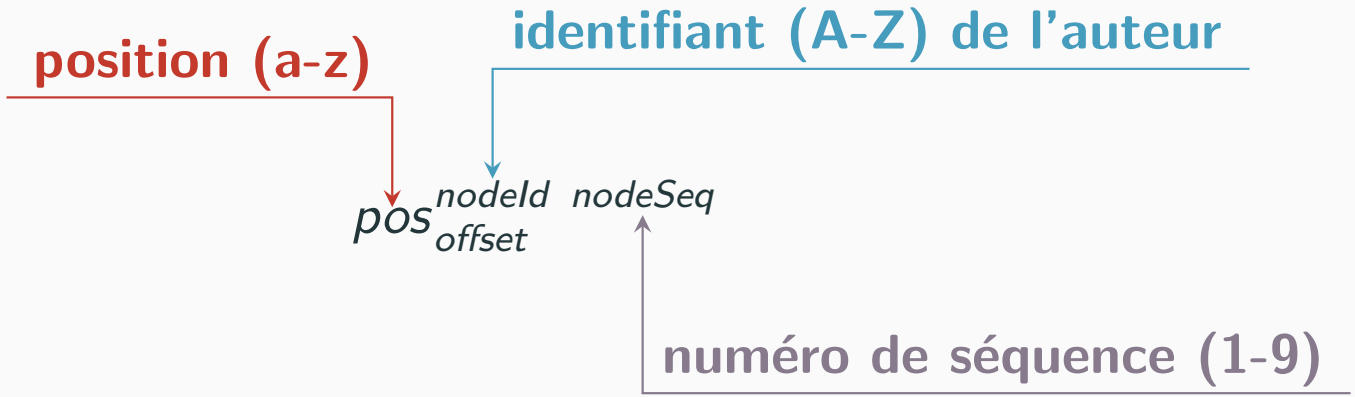
Exemples

$$d_0^{F5} <_{id} m_0^{C1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

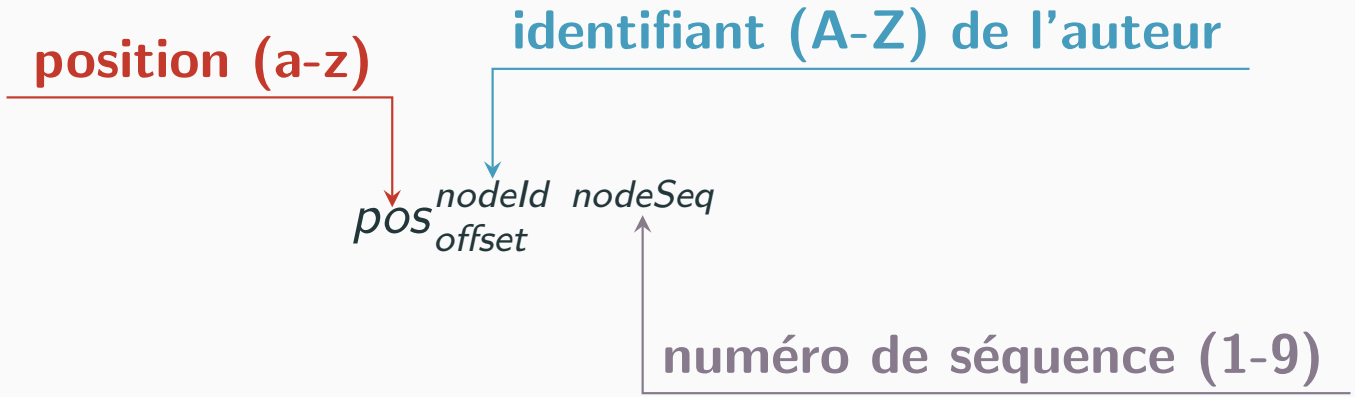
Exemples

$$\mathbf{d}_0^{F5} <_{id} \mathbf{m}_0^{C1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

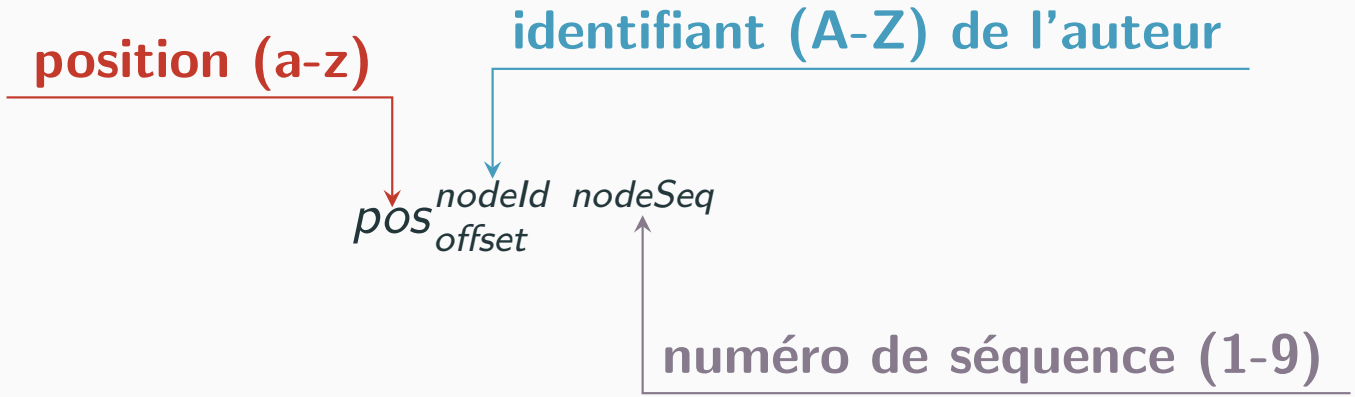
Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

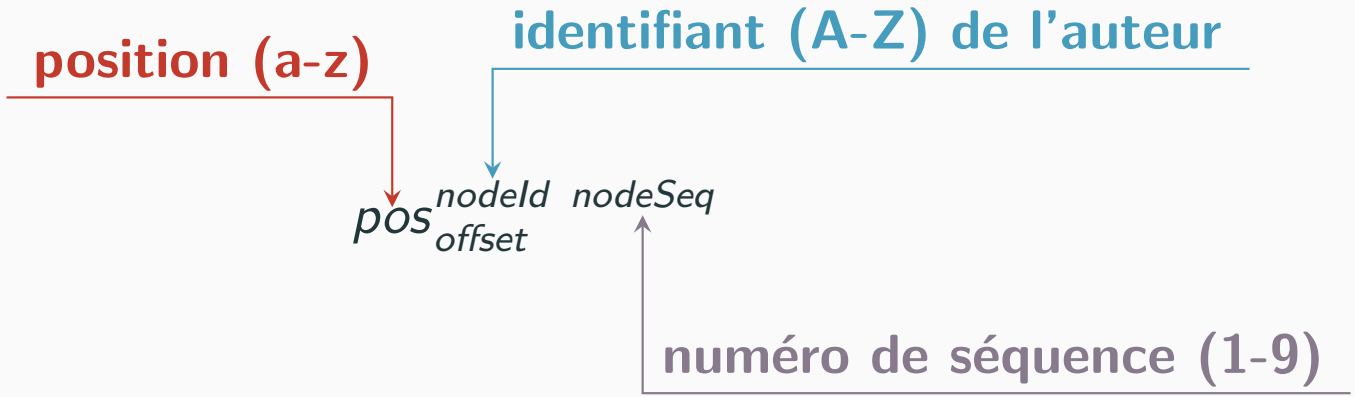
Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1}\mathbf{f_0^{E1}}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

Exemples

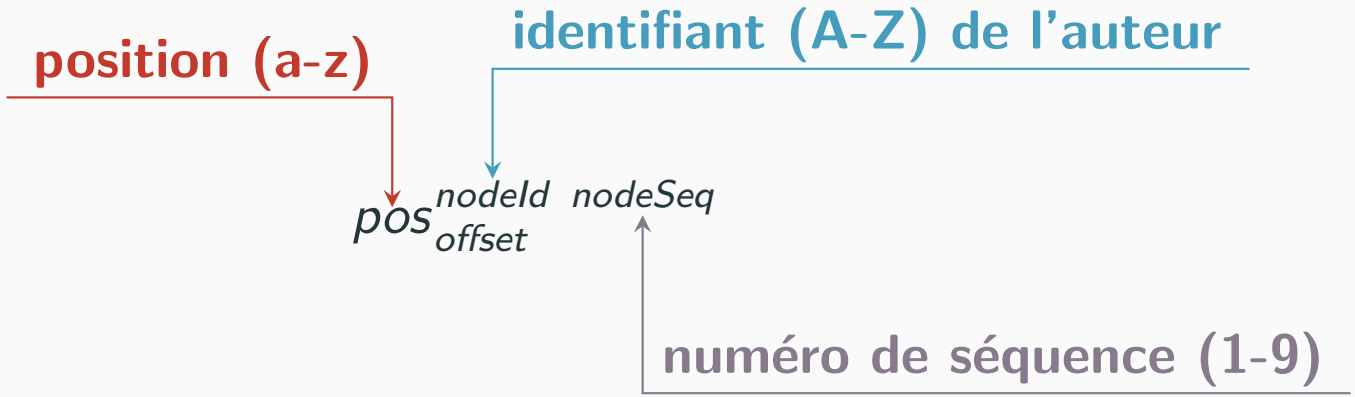
$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

$$i_0^{B1} <_{id} \quad ? \quad <_{id} i_1^{B1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Relation d'ordre $<_{id}$

- Se base sur l'ordre lexicographique sur les éléments des tuples

Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

$$i_0^{B1} <_{id} i_0^{B1} f_0^{A1} <_{id} i_1^{B1}$$

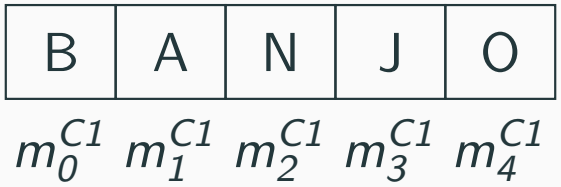
Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
m_0^{C1}	m_1^{C1}	m_2^{C1}	m_3^{C1}	m_4^{C1}

Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément



- Aggrège en un bloc éléments ayant identifiants contigus

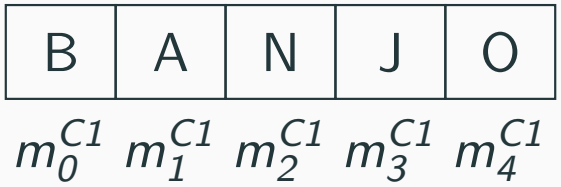
Identifiants contigus

Deux identifiants sont contigus si et seulement si :

1. les deux identifiants sont identiques à l'exception de leur dernier offset
2. ces deux derniers offsets sont consécutifs

Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément



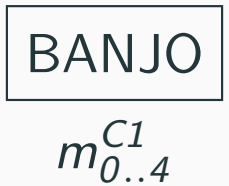
- Aggrège en un bloc éléments ayant identifiants contigus

Identifiants contigus

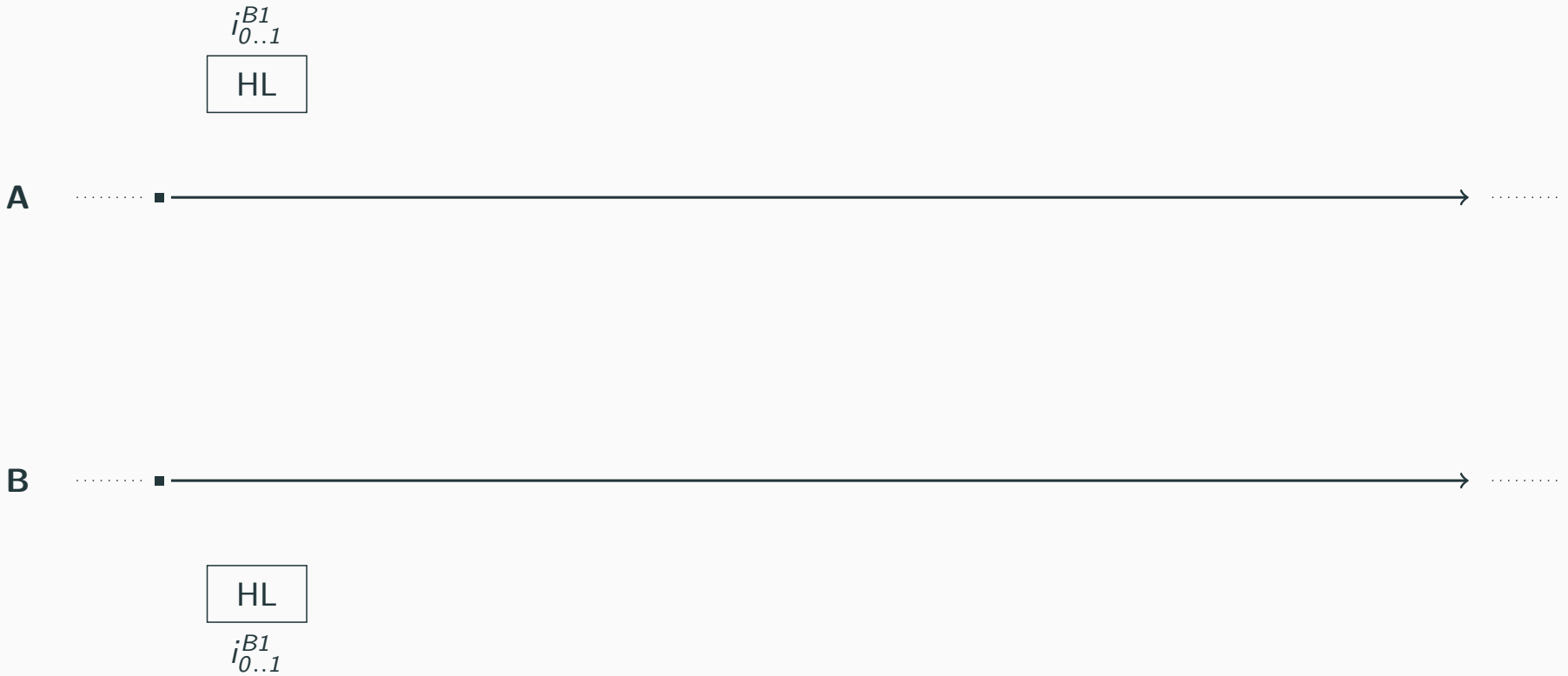
Deux identifiants sont contigus si et seulement si :

1. les deux identifiants sont identiques à l'exception de leur dernier offset
2. ces deux derniers offsets sont consécutifs

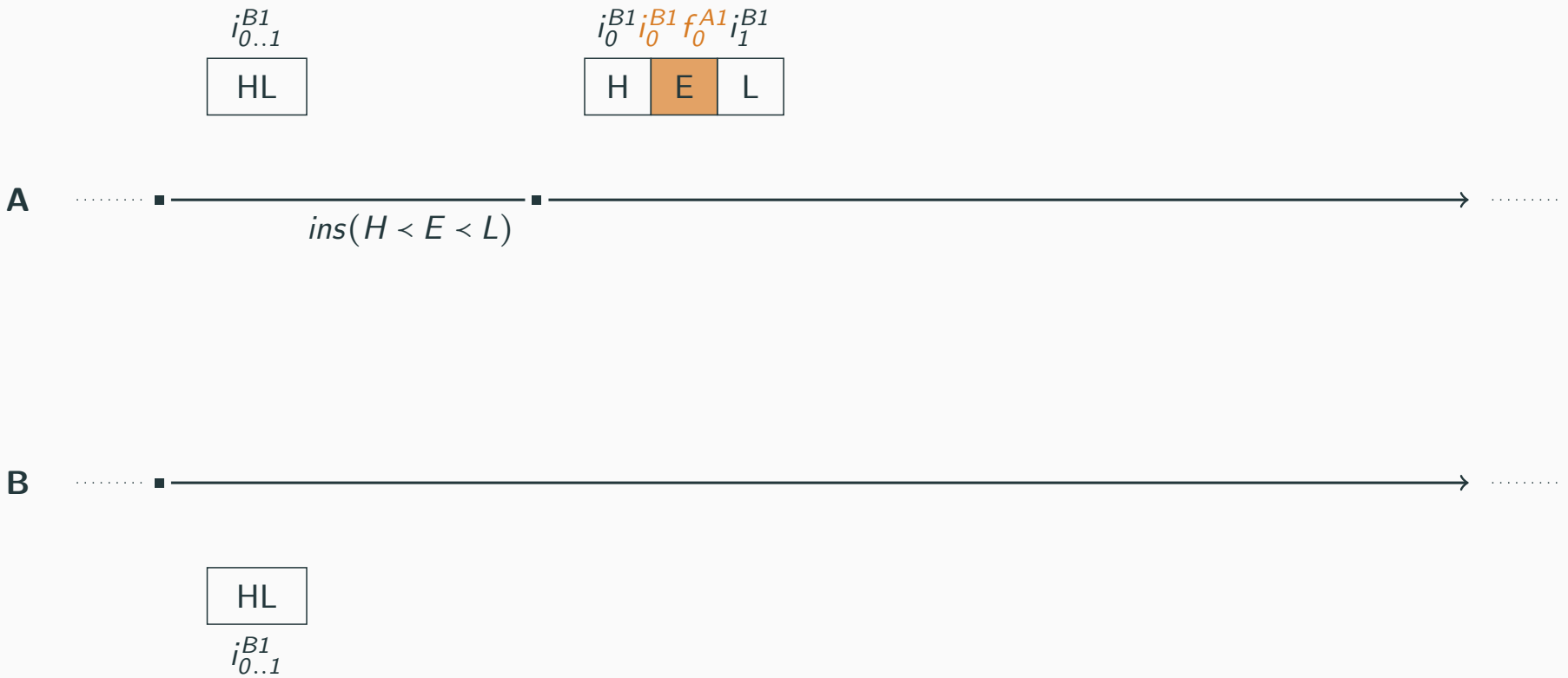
- Note l'intervalle d'identifiants d'un bloc : $pos_{begin..end}^{nodeId\ \ nodeSeq}$



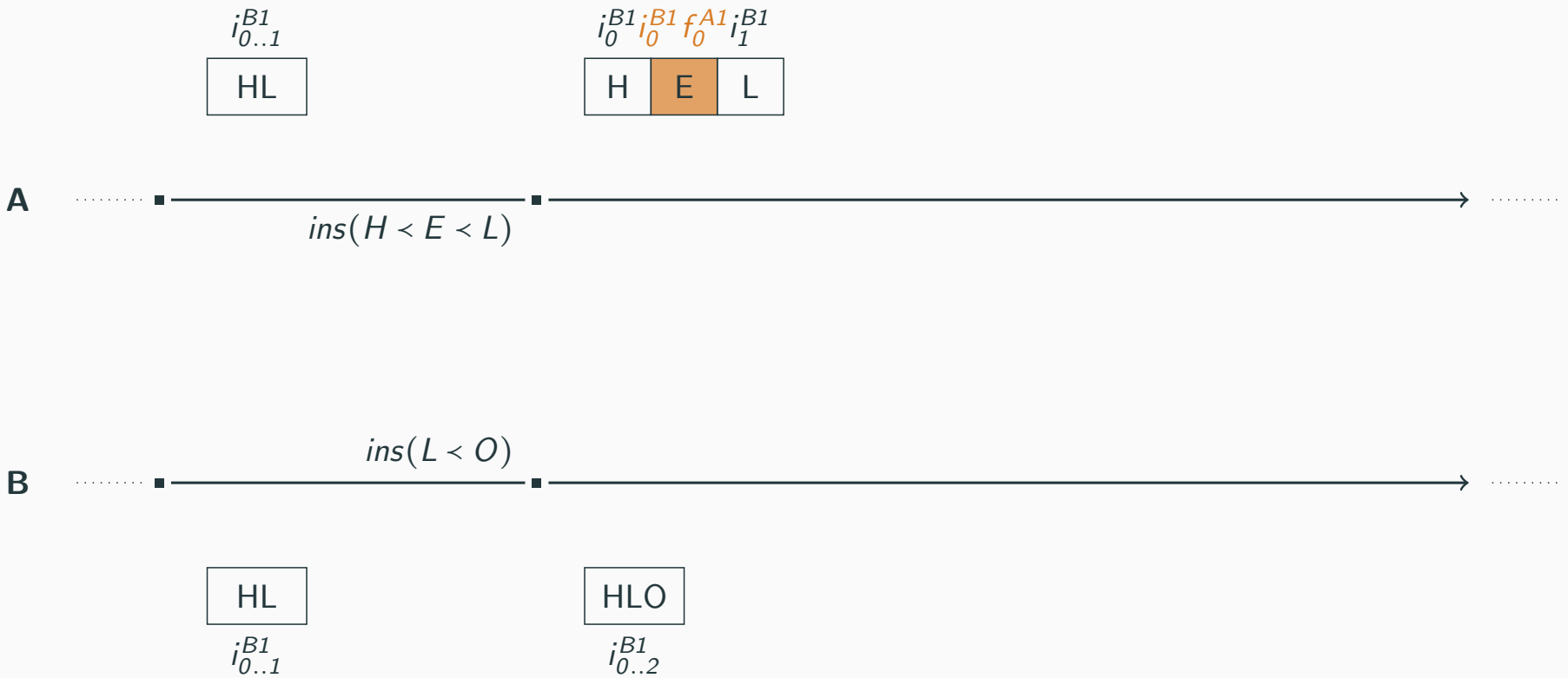
Exemple insertions concurrentes



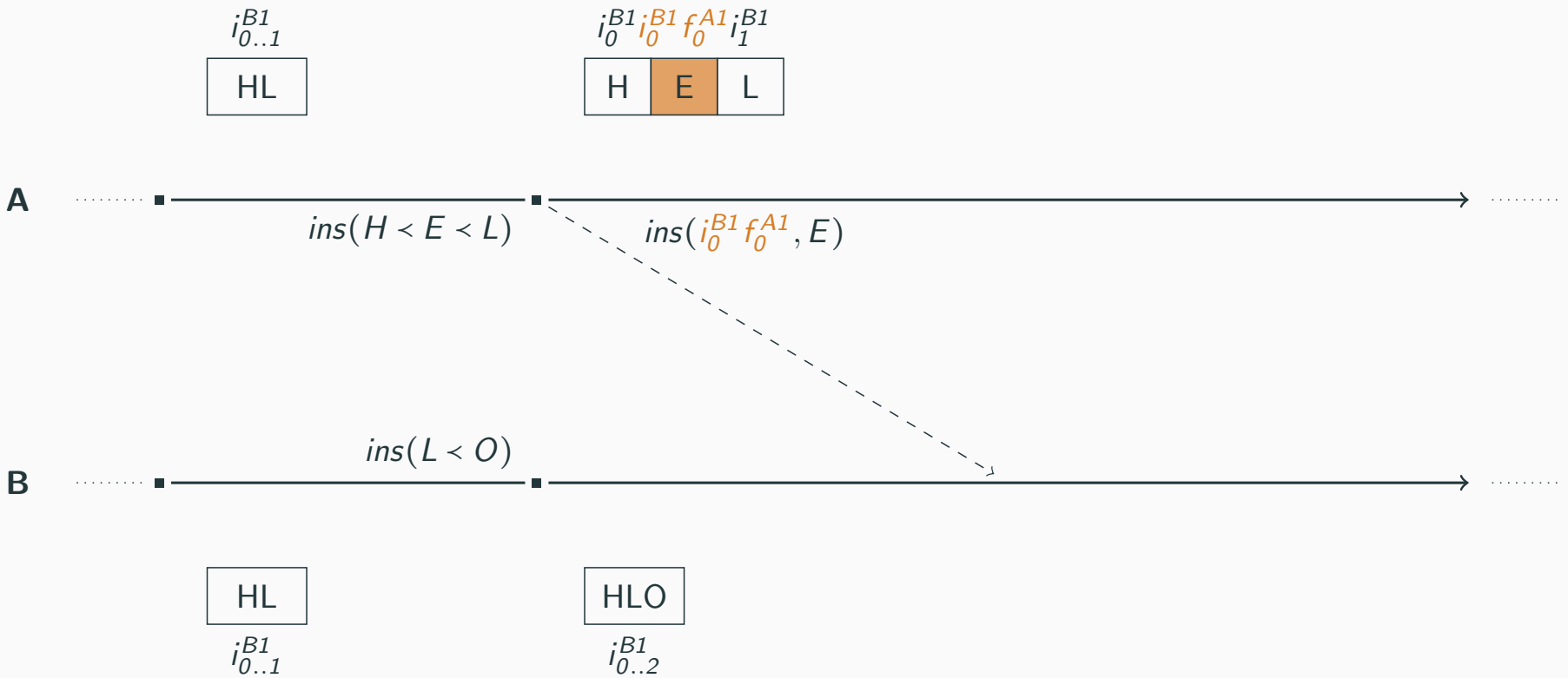
Exemple insertions concurrentes



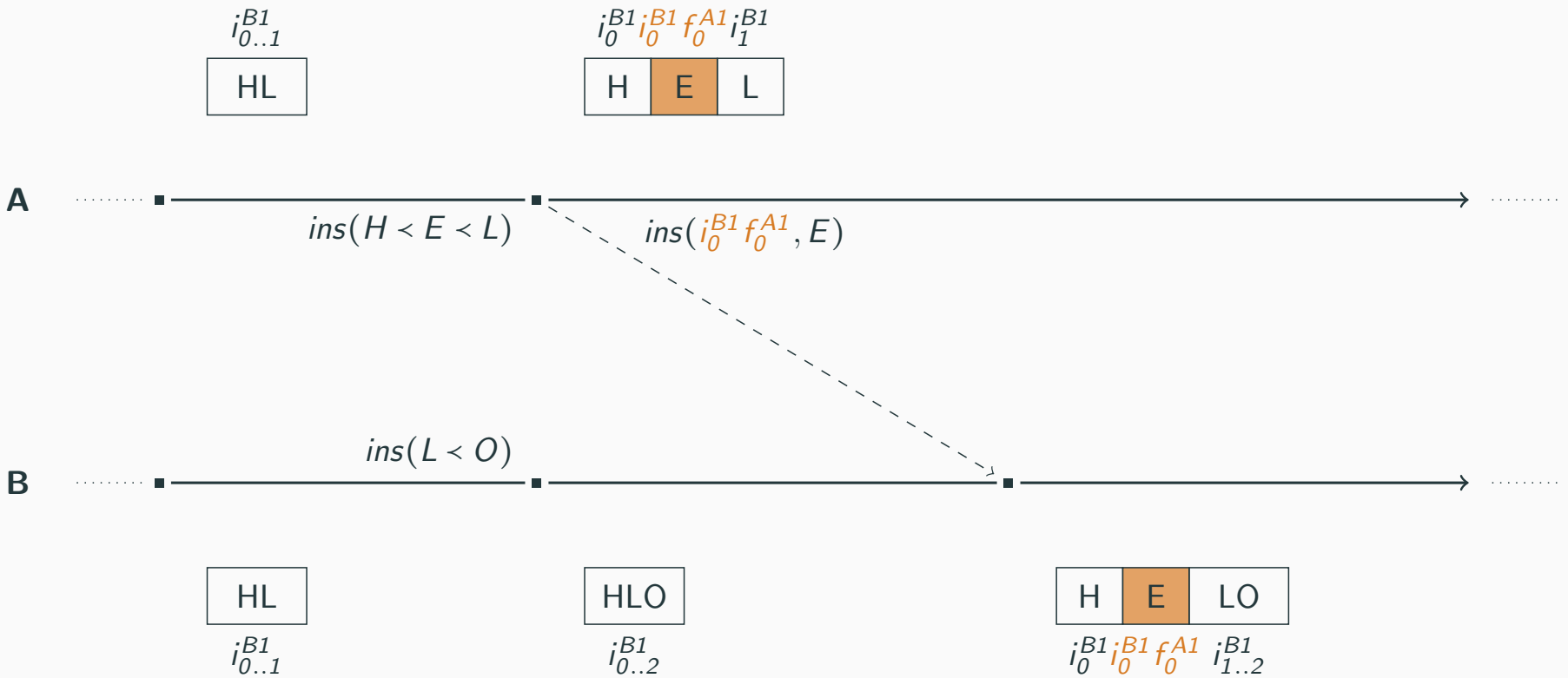
Exemple insertions concurrentes



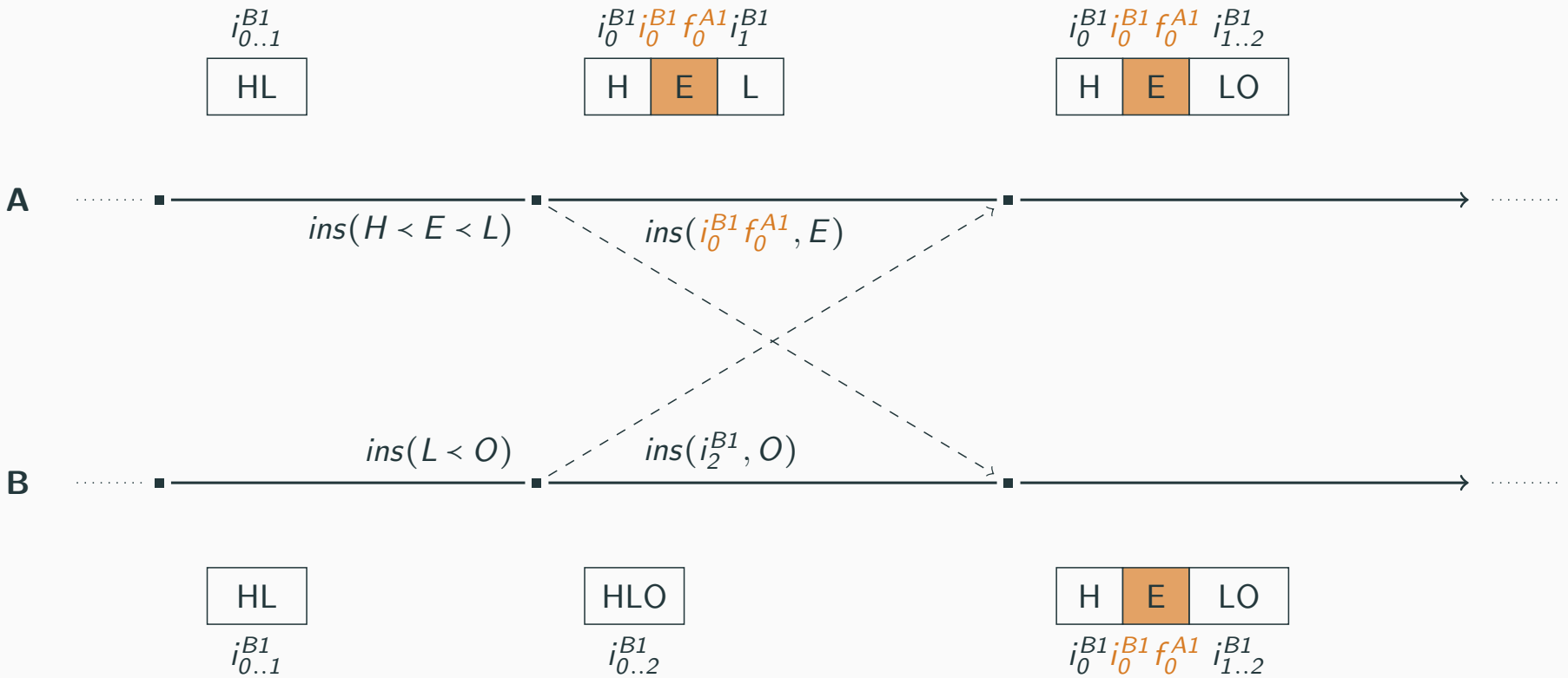
Exemple insertions concurrentes



Exemple insertions concurrentes



Exemple insertions concurrentes



Limites de LogootSplit

Sources de la croissance des métadonnées

- Augmentation non-bornée de la taille des identifiants
- Fragmentation de la séquence en un nombre croissant de blocs

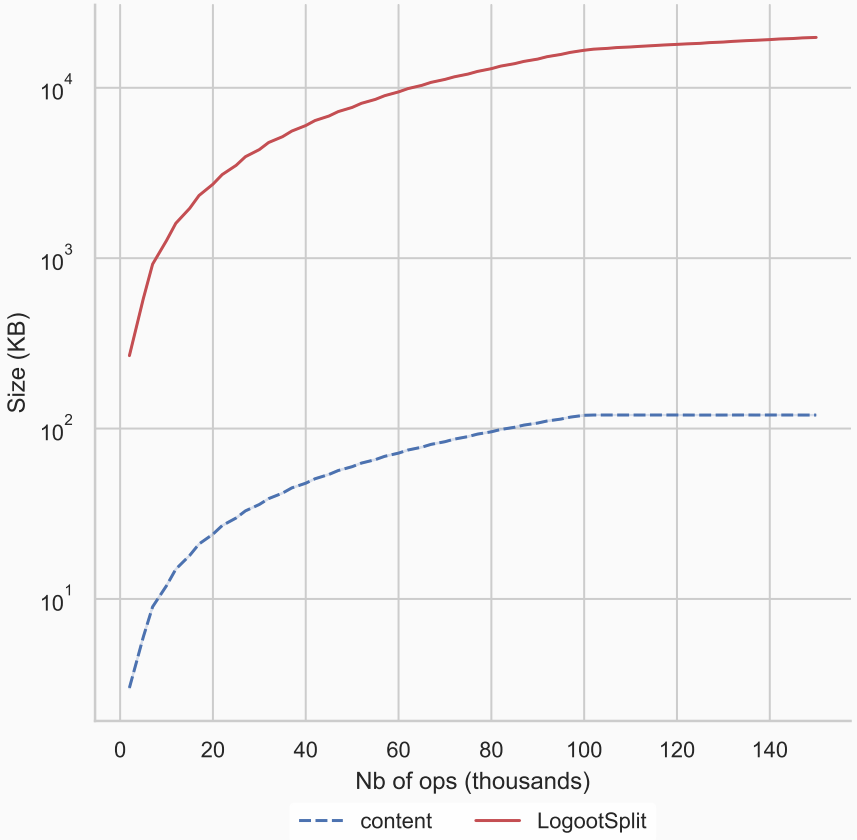
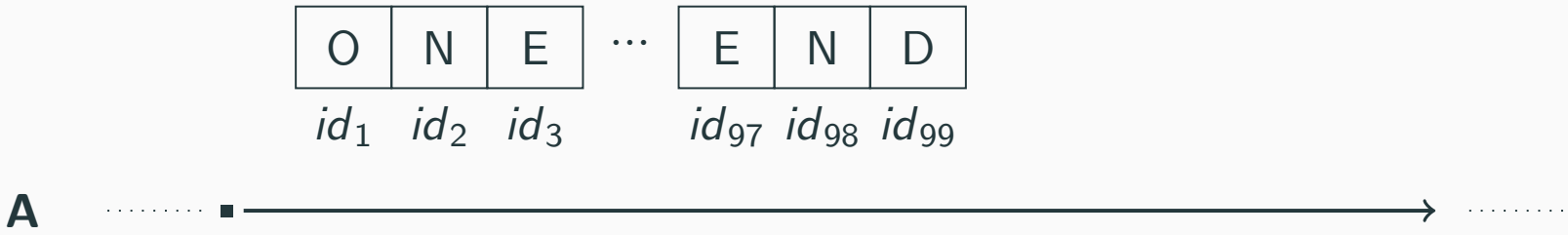


Figure 1 – Taille du contenu comparée à la taille de la séquence LogootSplit

Diminution des performances du point de vue mémoire, calculs et bande-passante

Comment réduire le surcoût ?

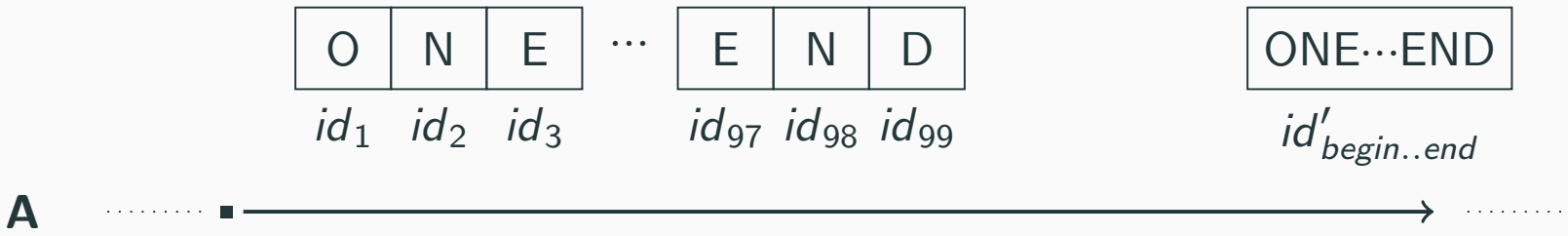
Solution naïve



- Convertir l'état actuel. . .

Comment réduire le surcoût ?

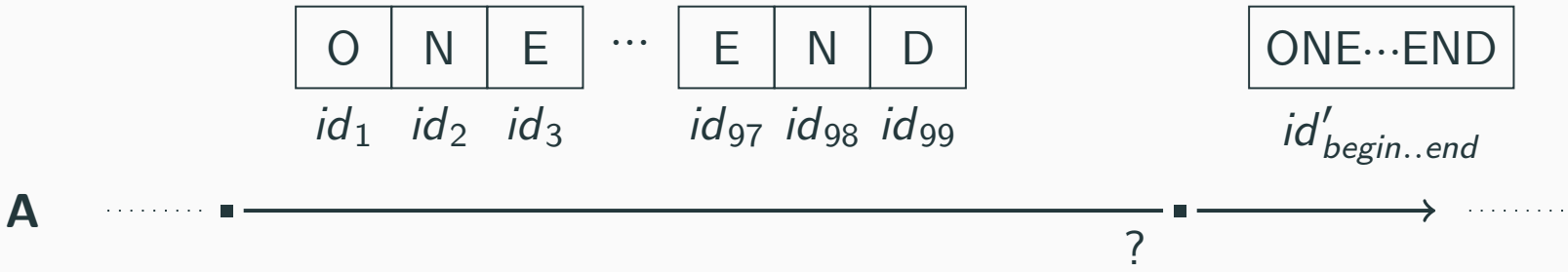
Solution naïve



- Convertir l'état actuel...
- ... en un état optimisé (identifiants de taille minimale, moins de blocs)...

Comment réduire le surcoût ?

Solution naïve



- Convertir l'état actuel...
- ... en un état optimisé (identifiants de taille minimale, moins de blocs)...
- ... à l'aide d'une nouvelle opération

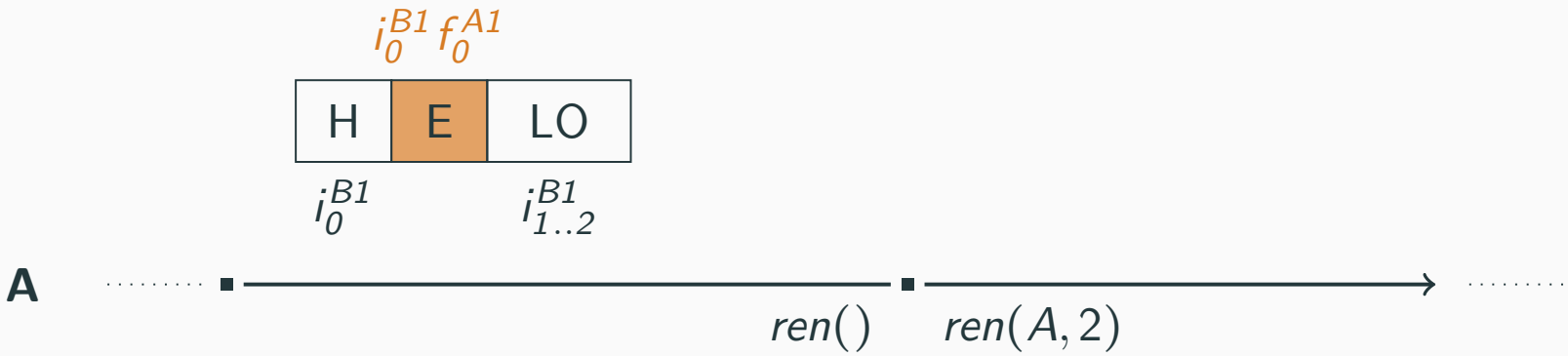
Contribution : RenamableLogootSplit

- CRDT pour le type Séquence qui incorpore un mécanisme de renommage
- Prend la forme d'une nouvelle opération : *rename*

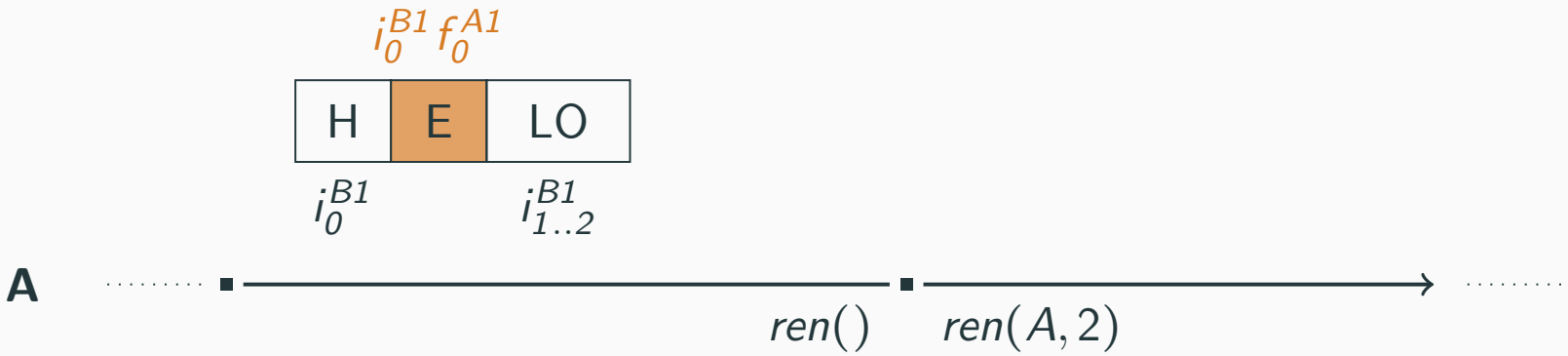
Propriétés de l'opération rename

- Est déterministe
- Préserve l'intention des utilisateur-rices
- Préserve les propriétés de la séquence, c.-à-d. l'unicité et l'ordre de ses identifiants
- Commute avec les opérations *insert*, *remove* mais aussi *rename* concurrentes

Opération rename

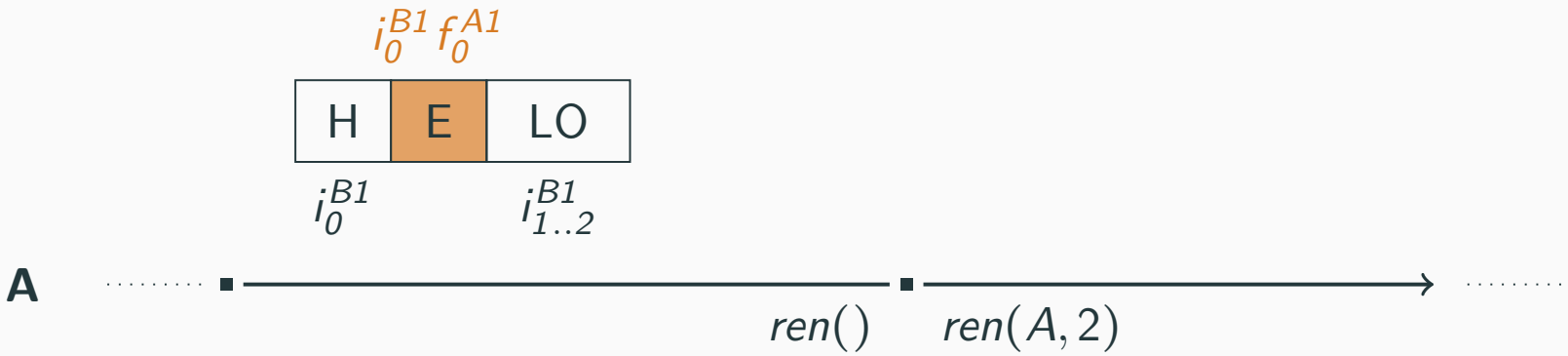


Opération rename



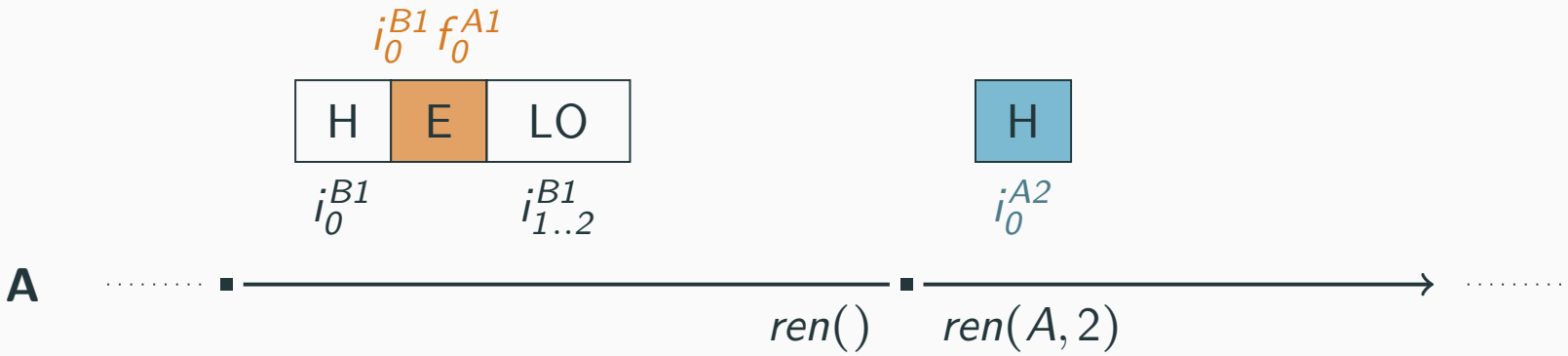
- Génère nouvel identifiant pour le 1er élément :

Opération rename



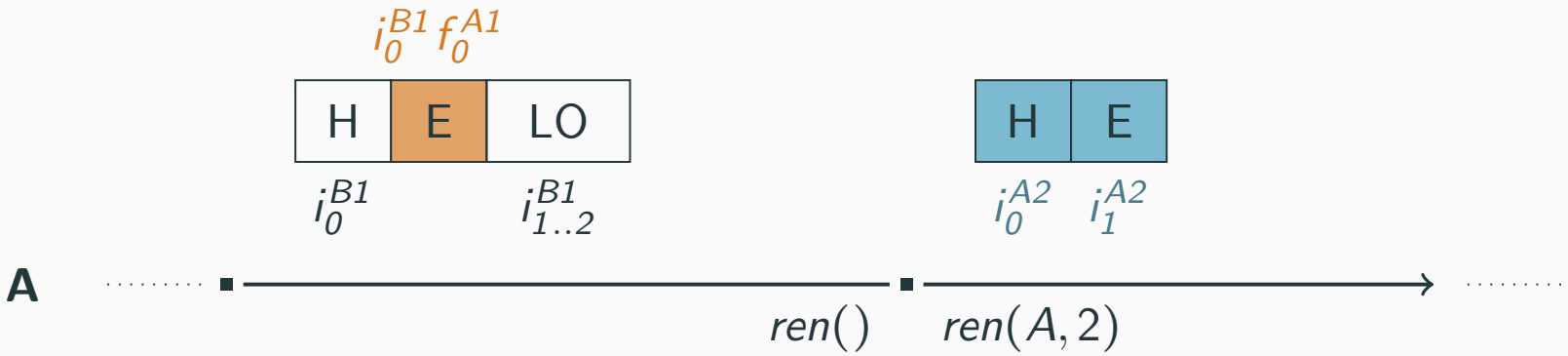
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$

Opération rename



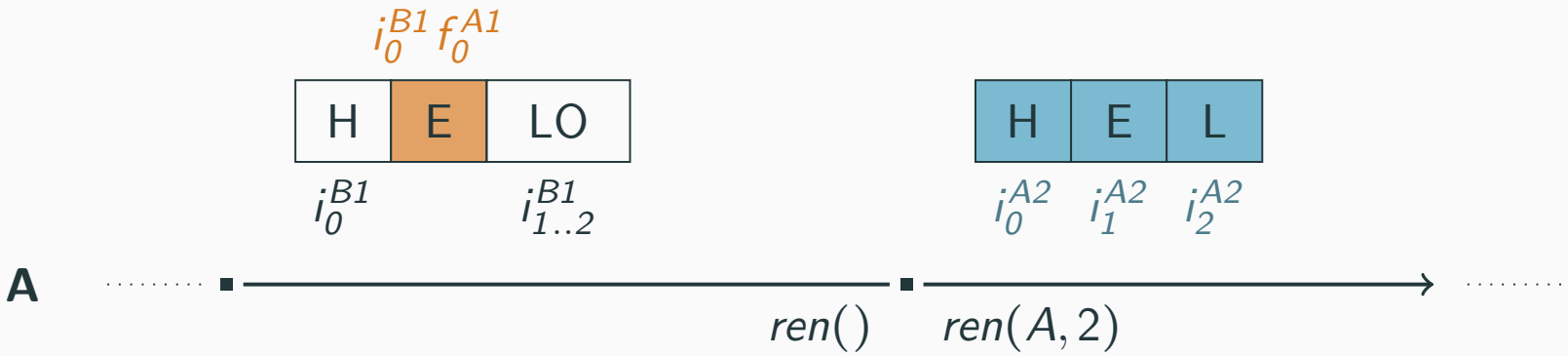
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :

Opération rename



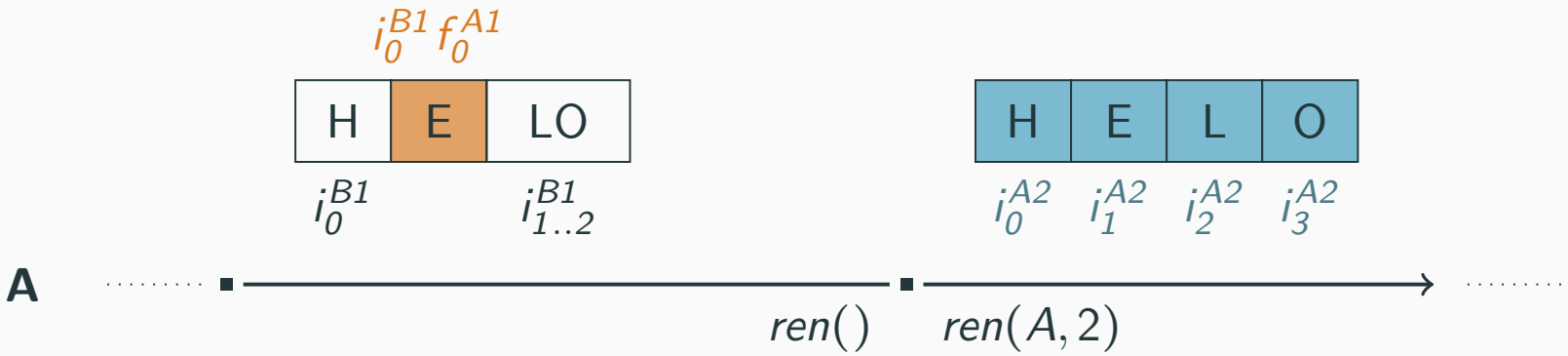
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}

Opération rename



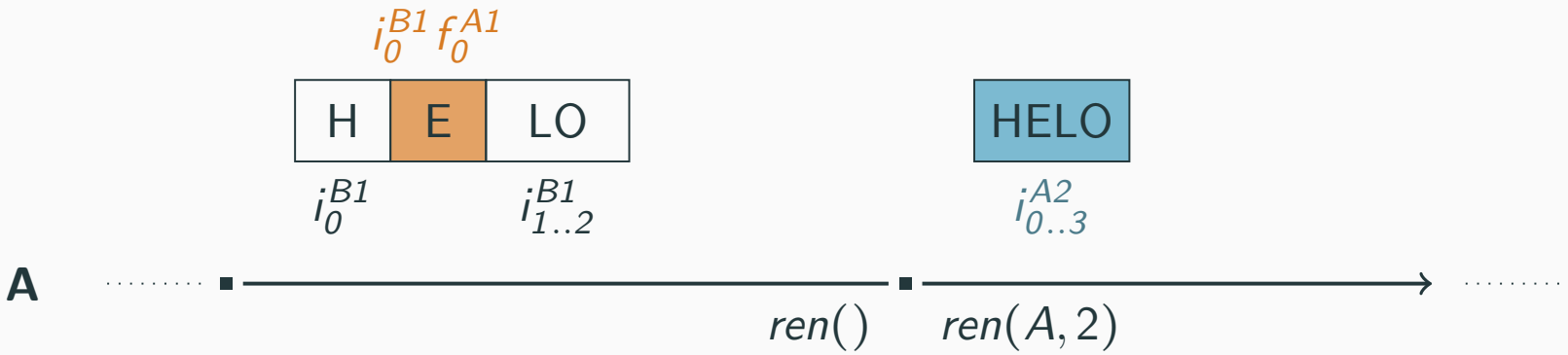
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}

Opération rename



- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}, ...

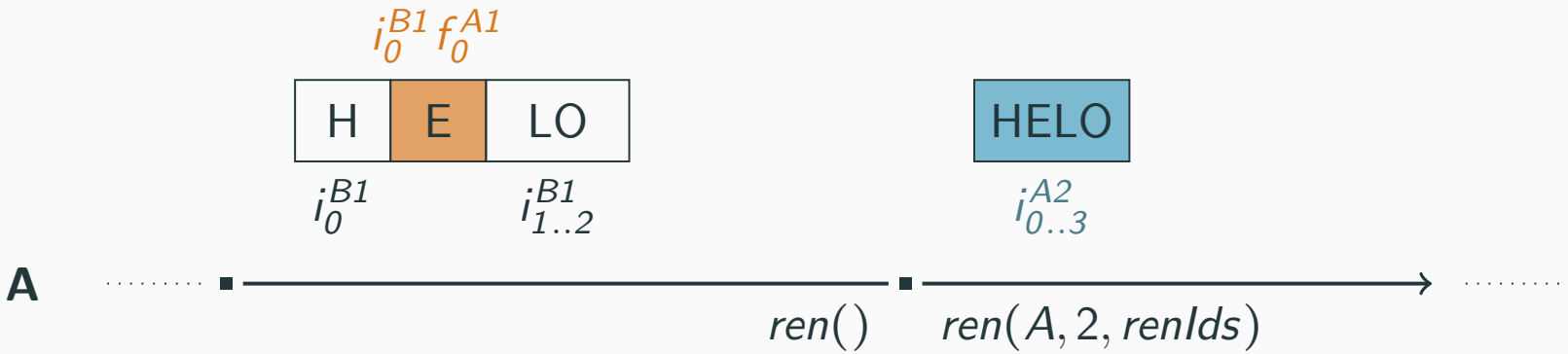
Opération rename



- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}, ...

Regroupe tous les éléments en 1 unique bloc

Opération rename



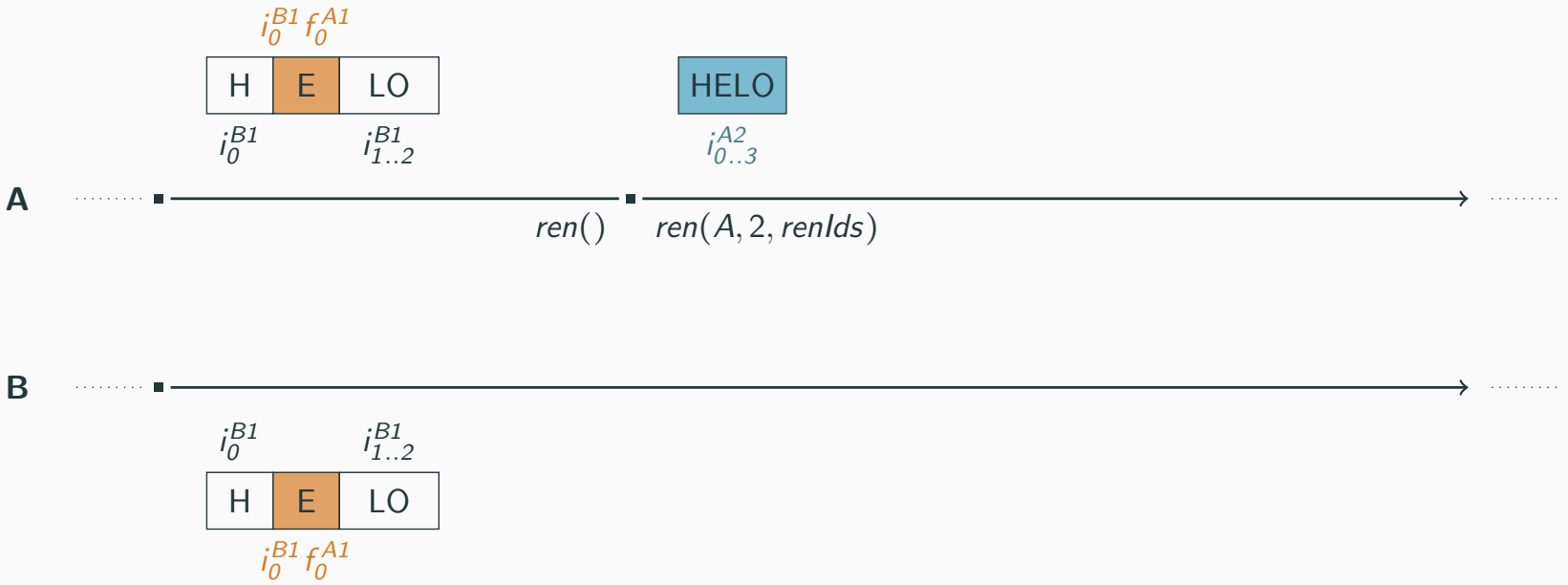
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}, ...

Regroupe tous les éléments en 1 unique bloc

Pour plus tard :

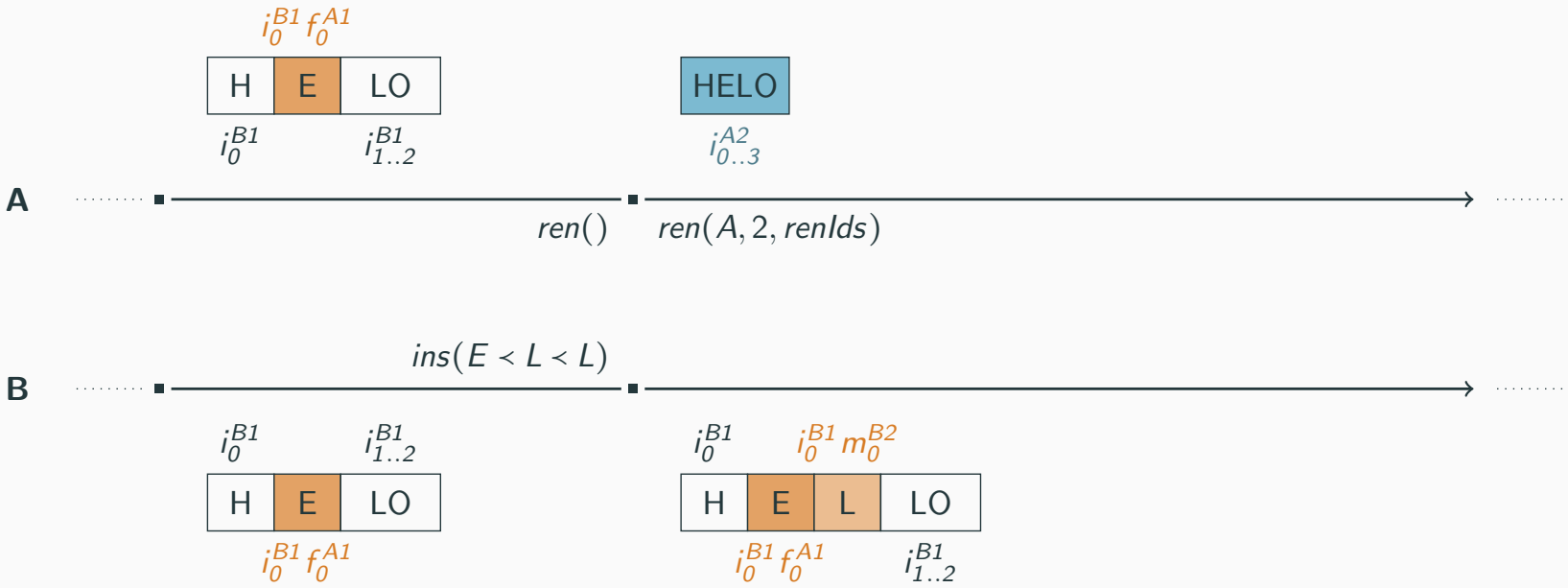
- Stocke identifiants ($[i_0^{B1}, i_0^{B1}\, f_0^{A1}, \ldots]$) de l'état d'origine : *renIds*

Interactions avec opérations insert et remove concurrentes



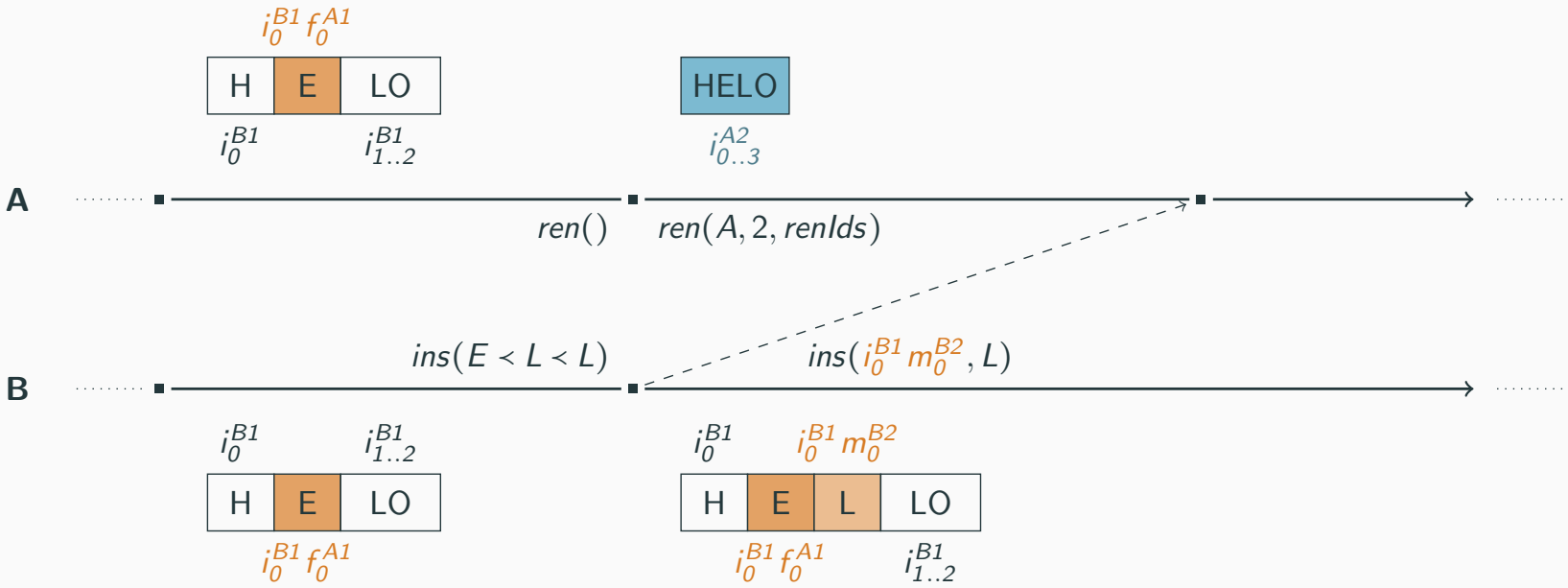
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations insert et remove concurrentes



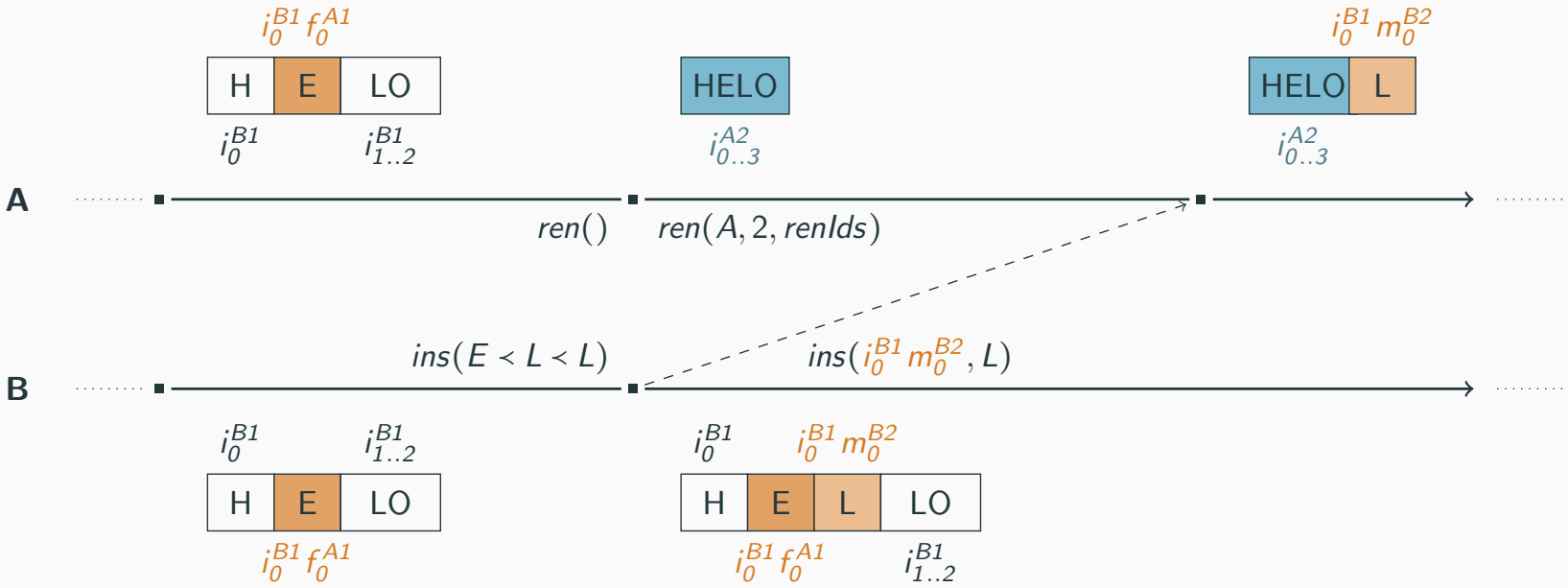
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations insert et remove concurrentes



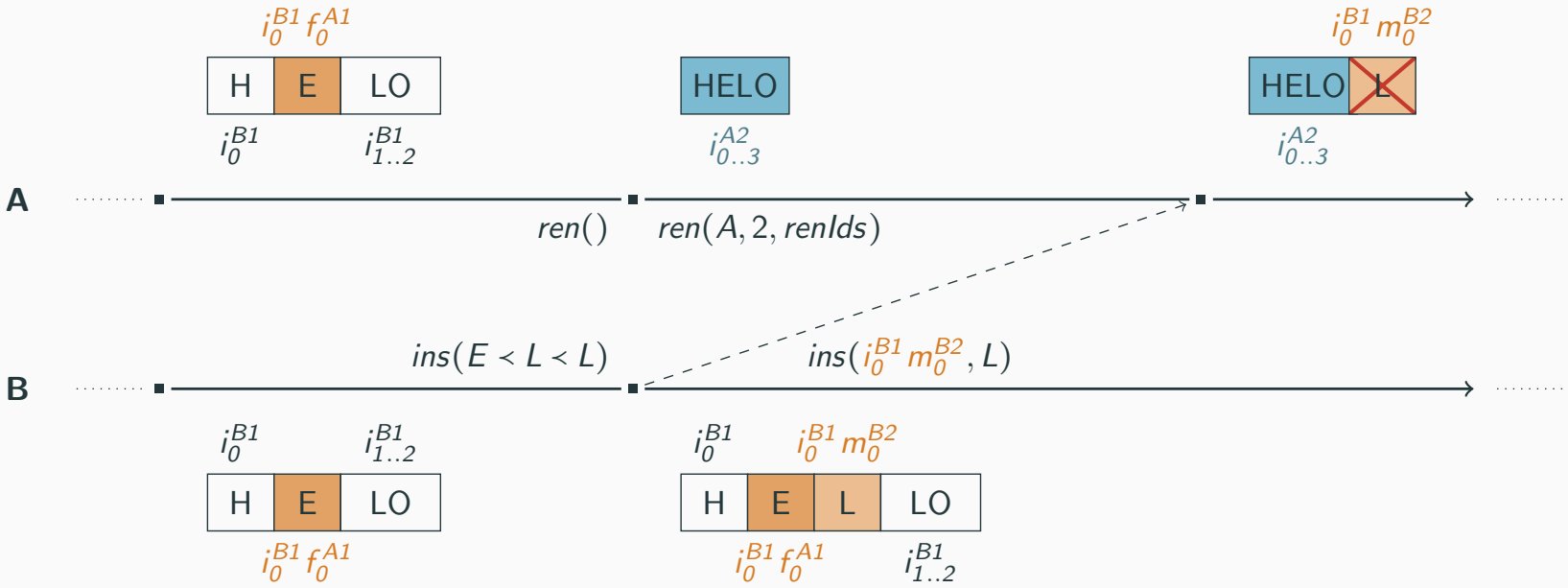
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations insert et remove concurrentes



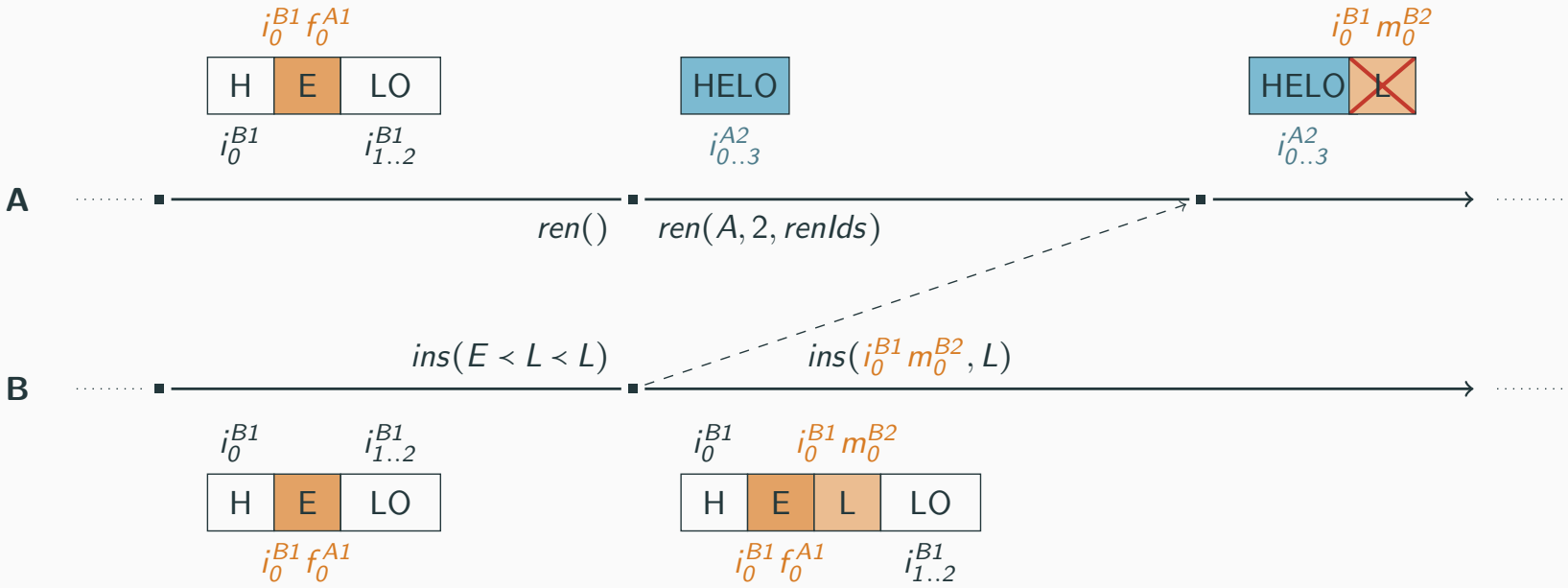
- Noeuds peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations insert et remove concurrentes



- Noeuds peuvent générer opérations concurrentes aux opérations *rename*
- Opérations produisent anomalies si intégrées naïvement

Interactions avec opérations insert et remove concurrentes



- Noeuds peuvent générer opérations concurrentes aux opérations *rename*
- Opérations produisent anomalies si intégrées naïvement

Nécessité d'un mécanisme dédié

Mécanisme de résolution de conflits entre une opération rename et une opération insert ou remove

Besoins

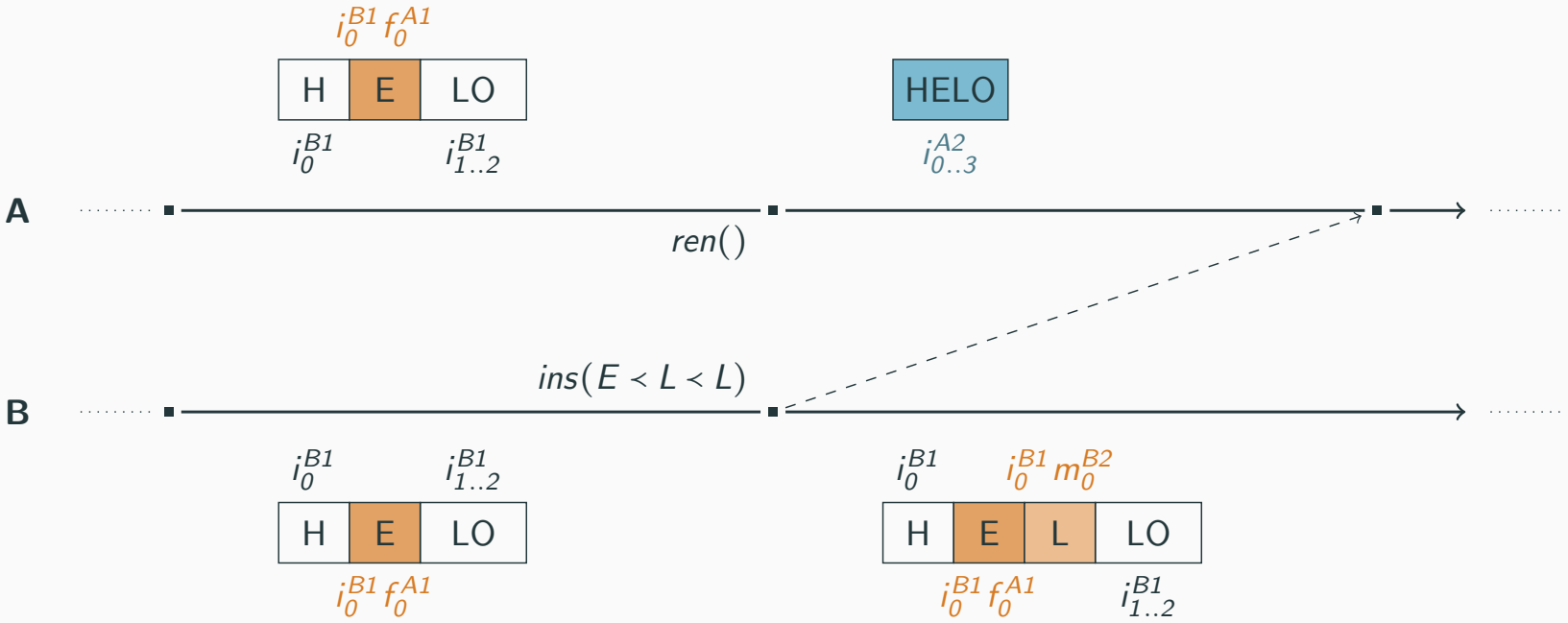
1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

Mécanisme de résolution de conflits entre une opération rename et une opération insert ou remove

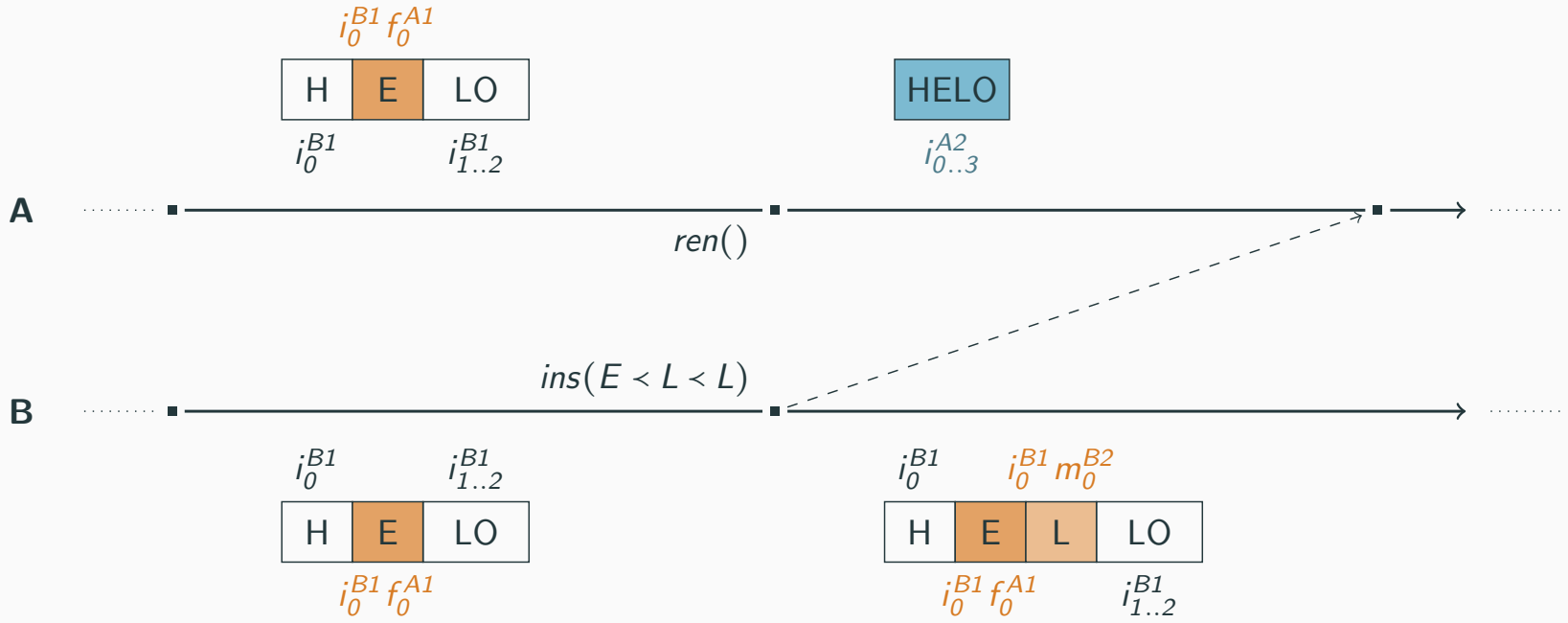
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

Détection des opérations concurrentes à opération rename

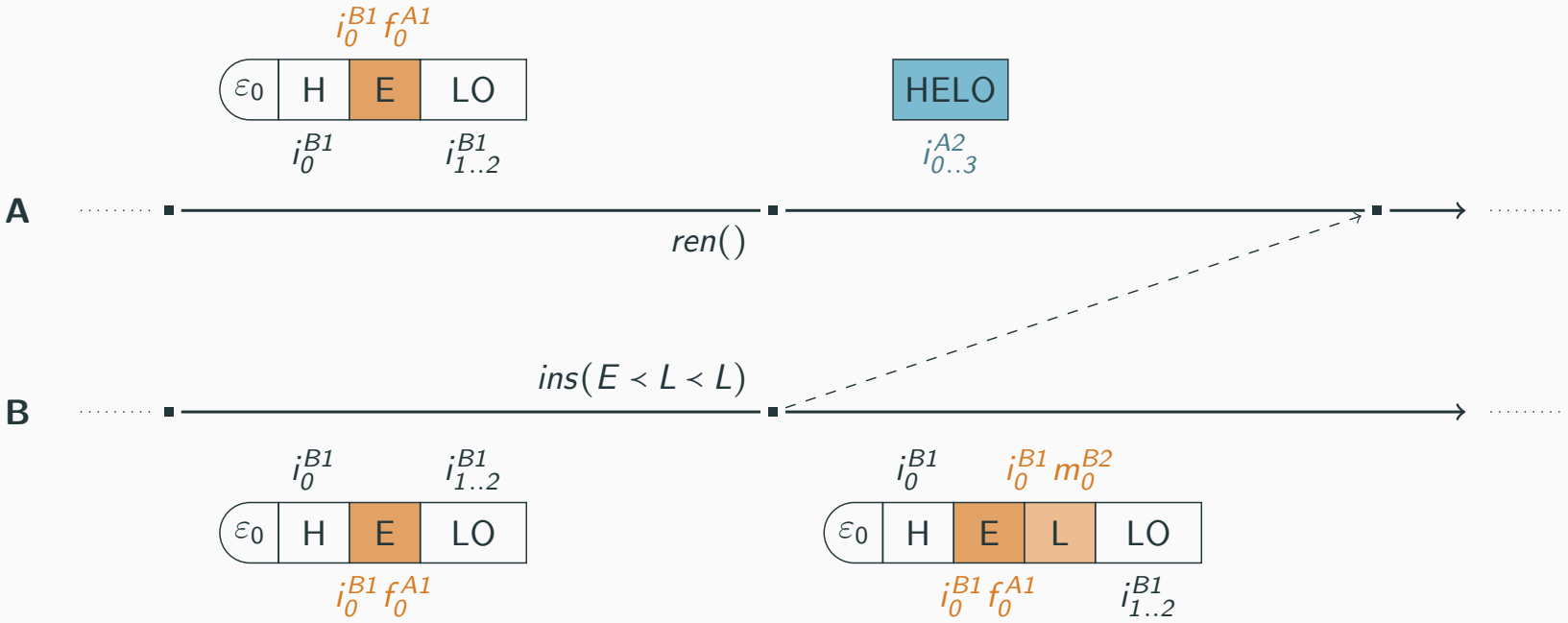


Détection des opérations concurrentes à opération rename



Ajout mécanisme d'époques

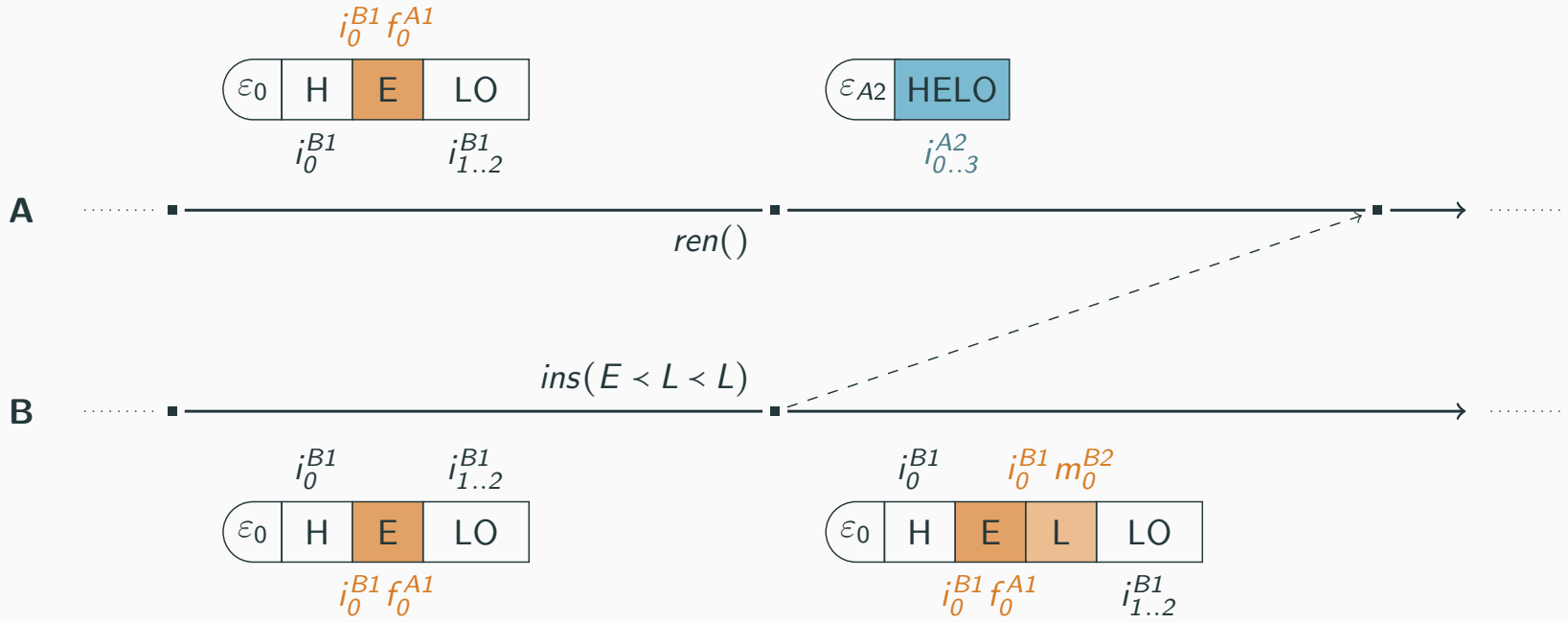
Détection des opérations concurrentes à opération rename



Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée ε_0

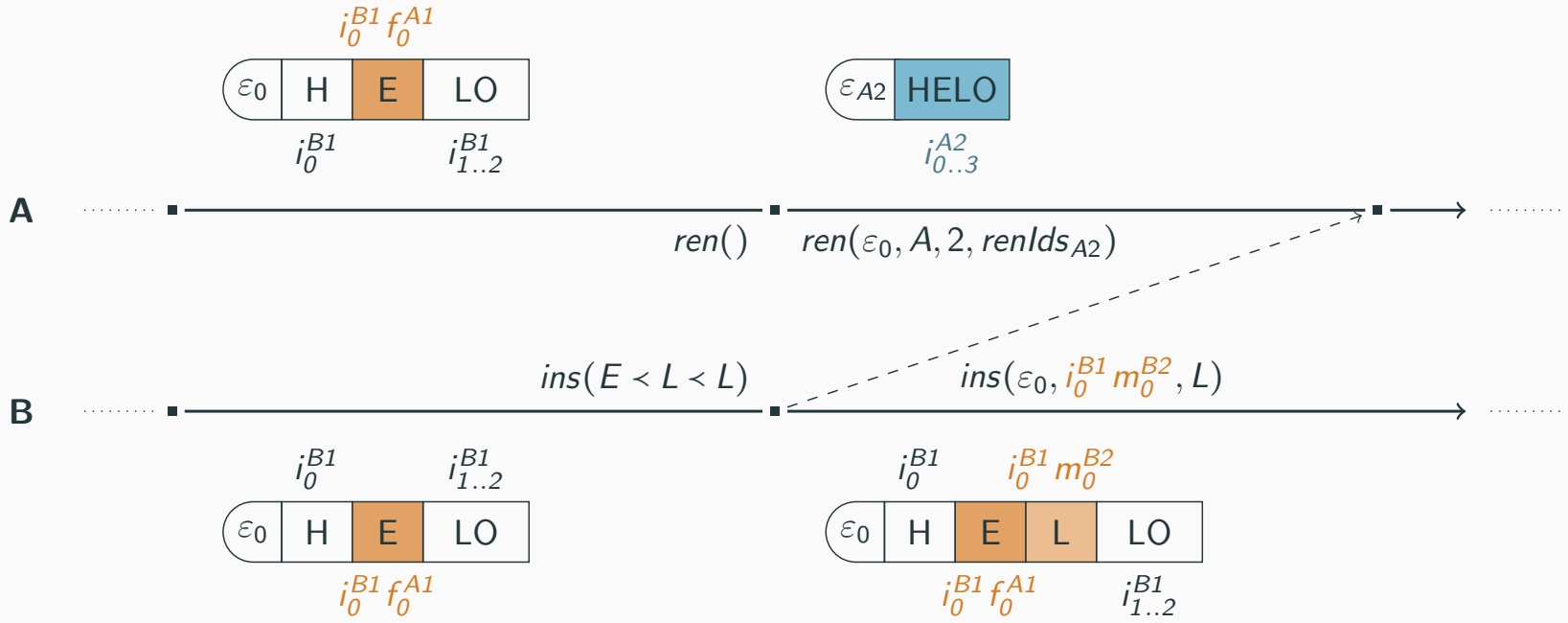
Détection des opérations concurrentes à opération rename



Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée ε_0
- *rename* font progresser à nouvelle époque, $\varepsilon_{nodeId\ \ nodeSeq}$

Détection des opérations concurrentes à opération rename



Ajout mécanisme d'époques

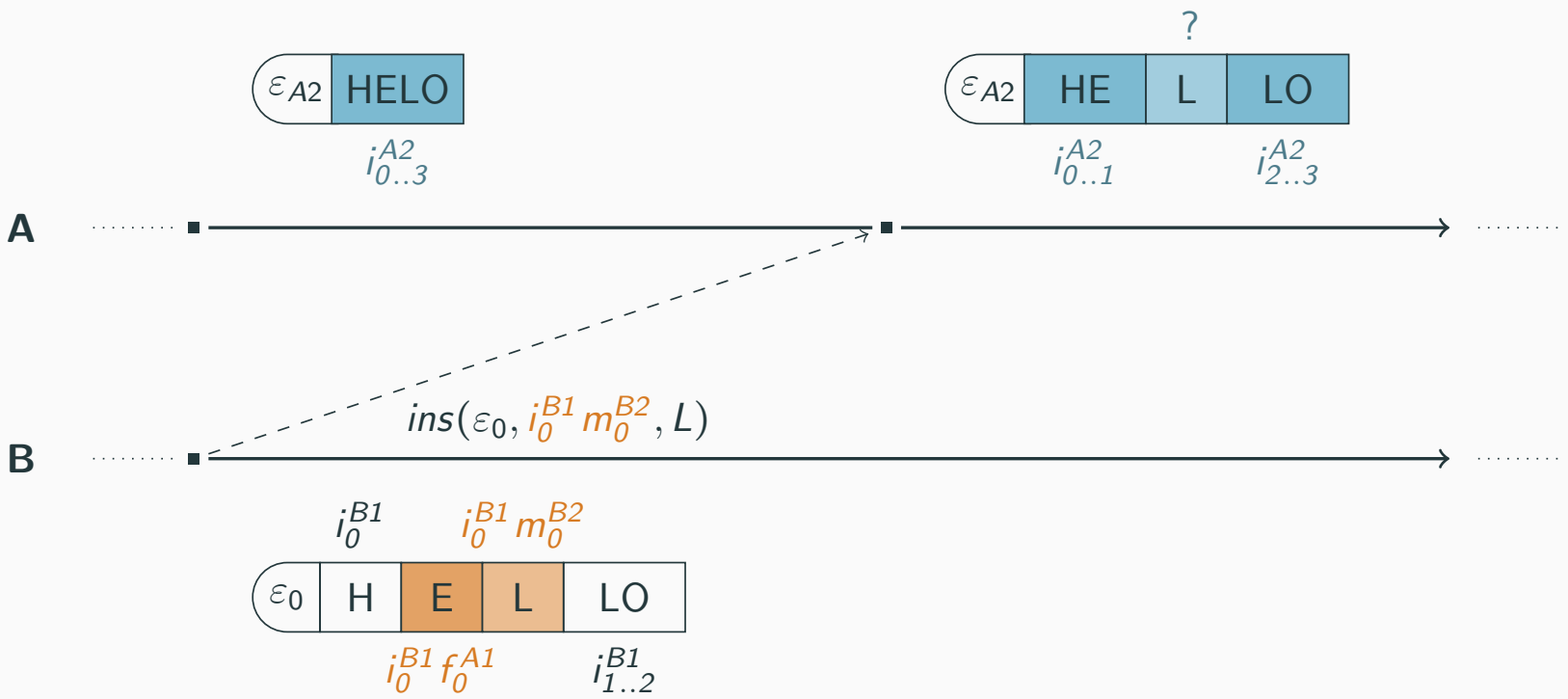
- Séquence commence à époque d'origine, notée ε_0
- *rename* font progresser à nouvelle époque, $\varepsilon_{nodeId\ \ nodeSeq}$
- Opérations labellisées avec époque de génération

Mécanisme de résolution de conflits

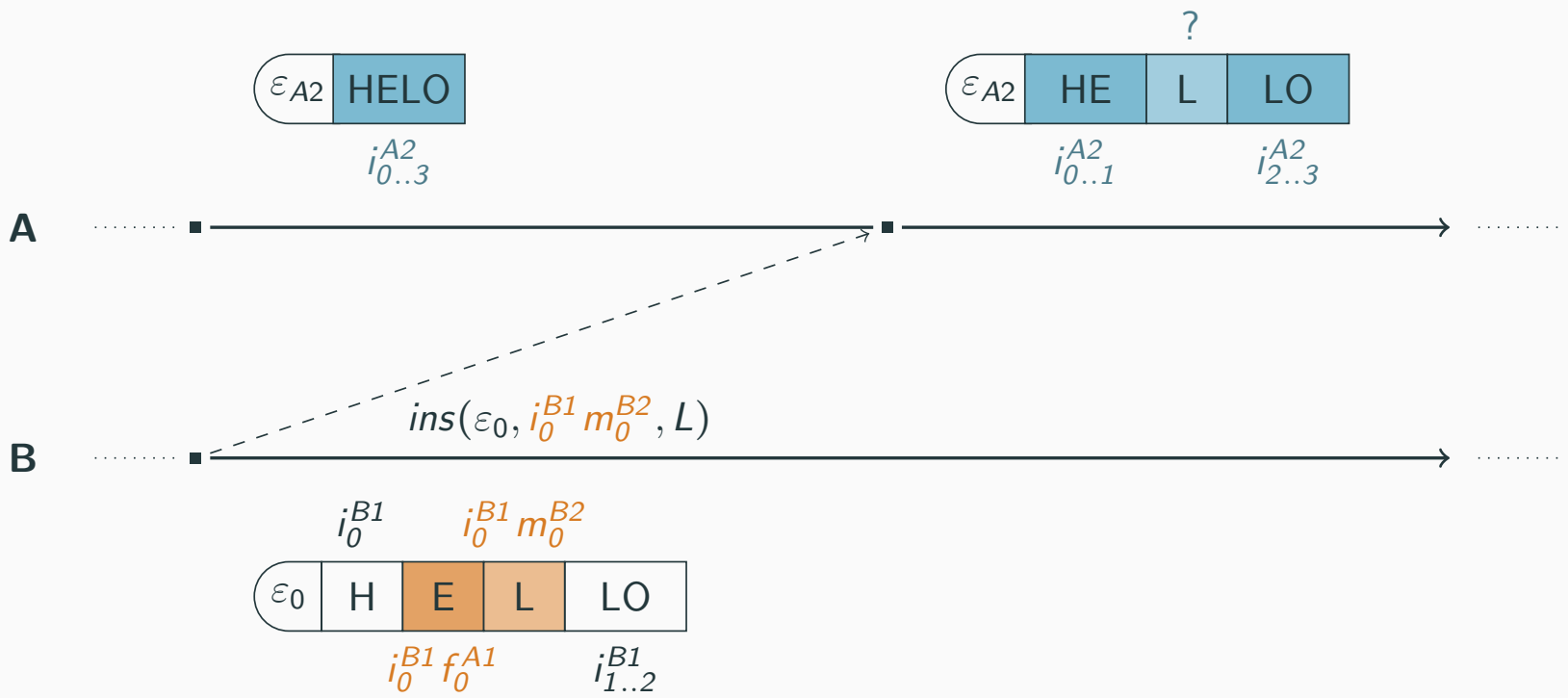
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

Intégration des opérations insert et remove concurrentes

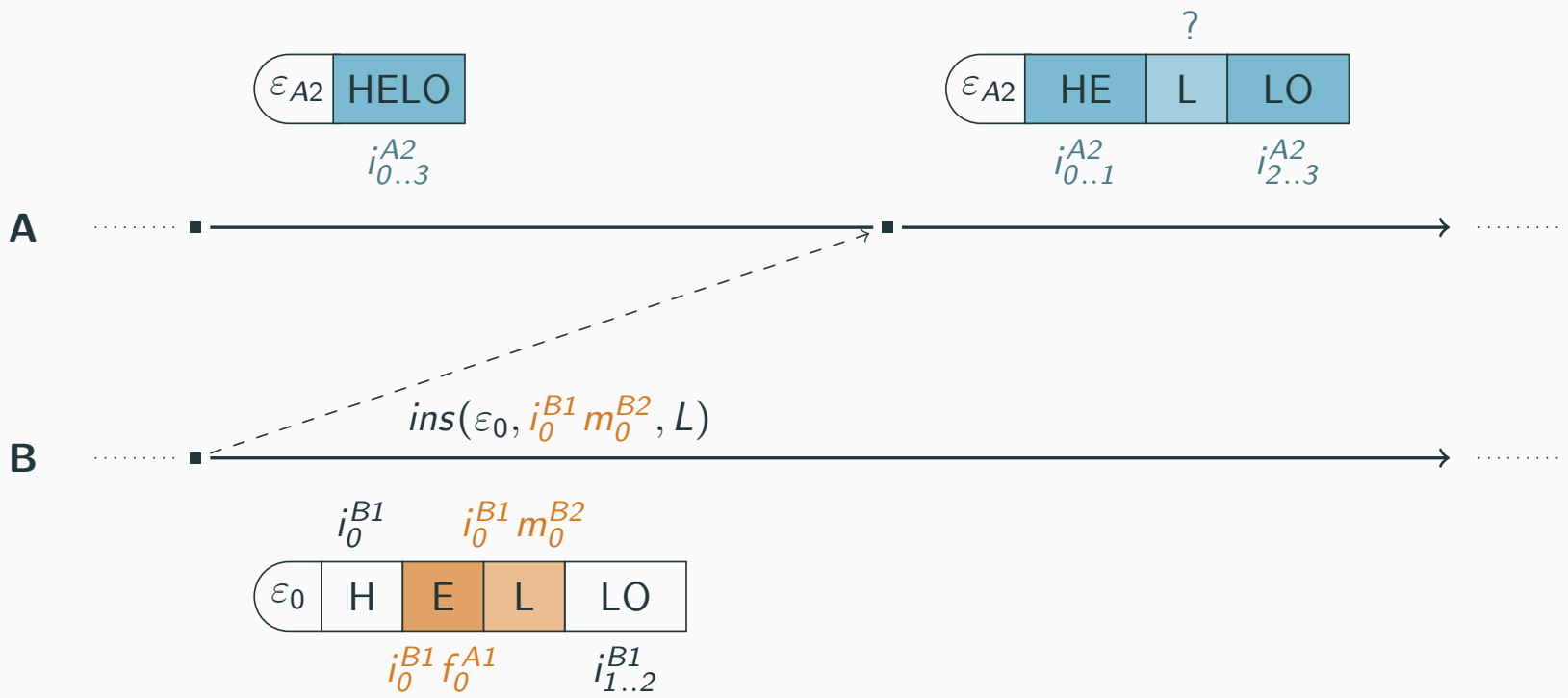


Intégration des opérations insert et remove concurrentes



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

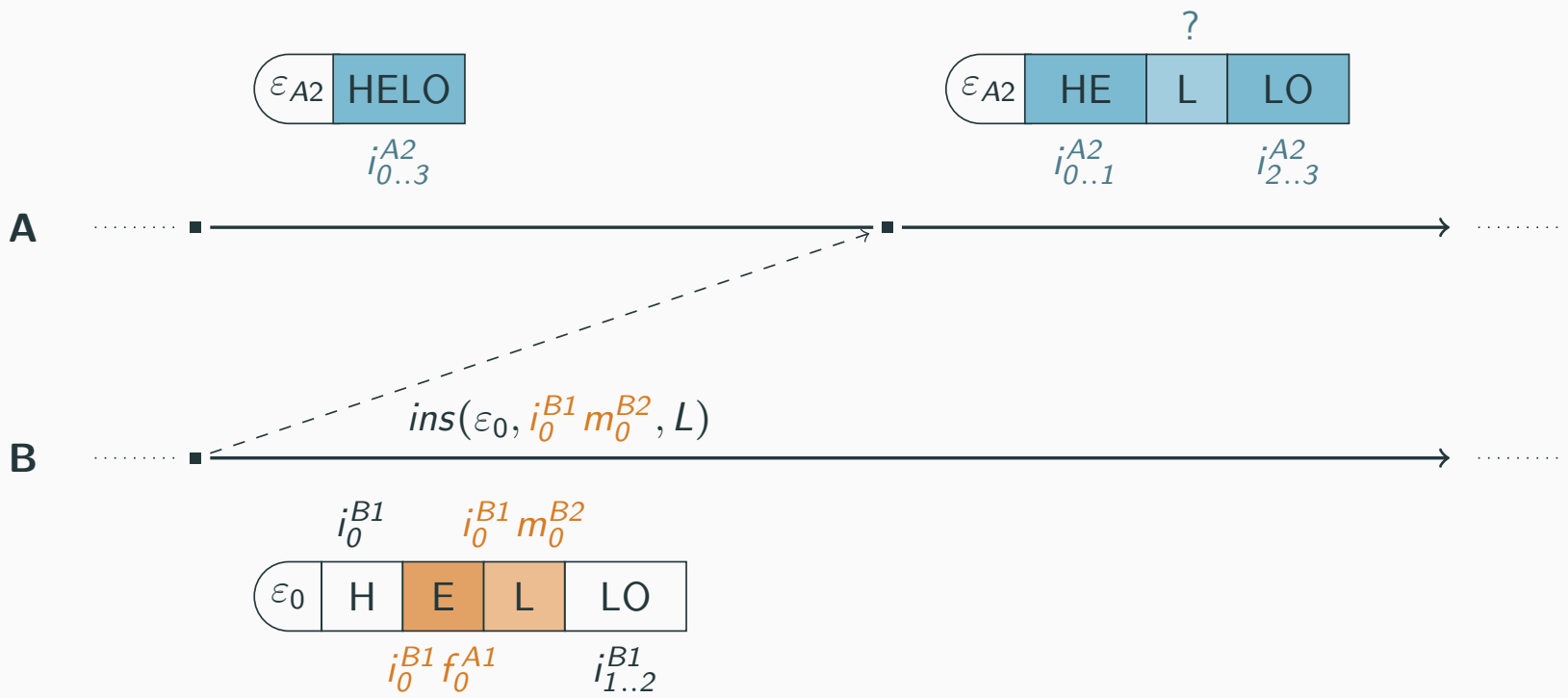
Intégration des opérations insert et remove concurrentes



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

- Prend la forme de l'algorithme `renameId`

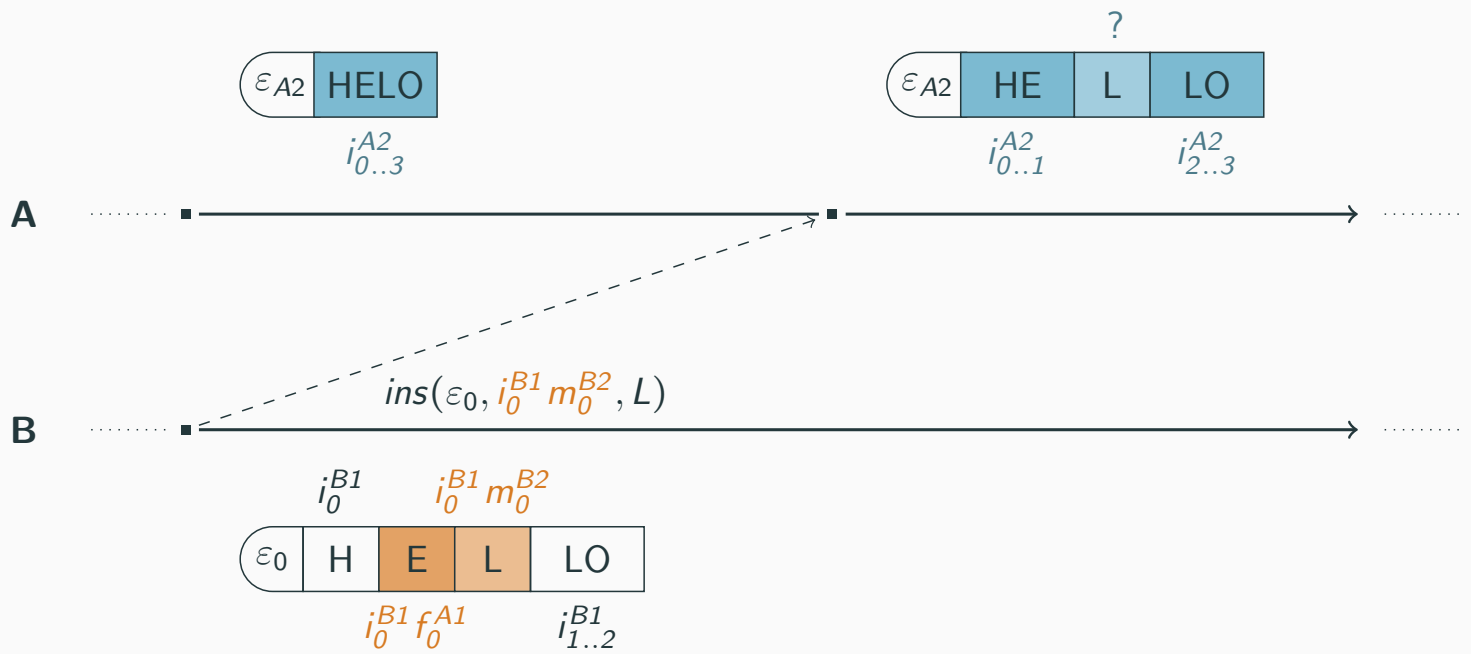
Intégration des opérations insert et remove concurrentes



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`

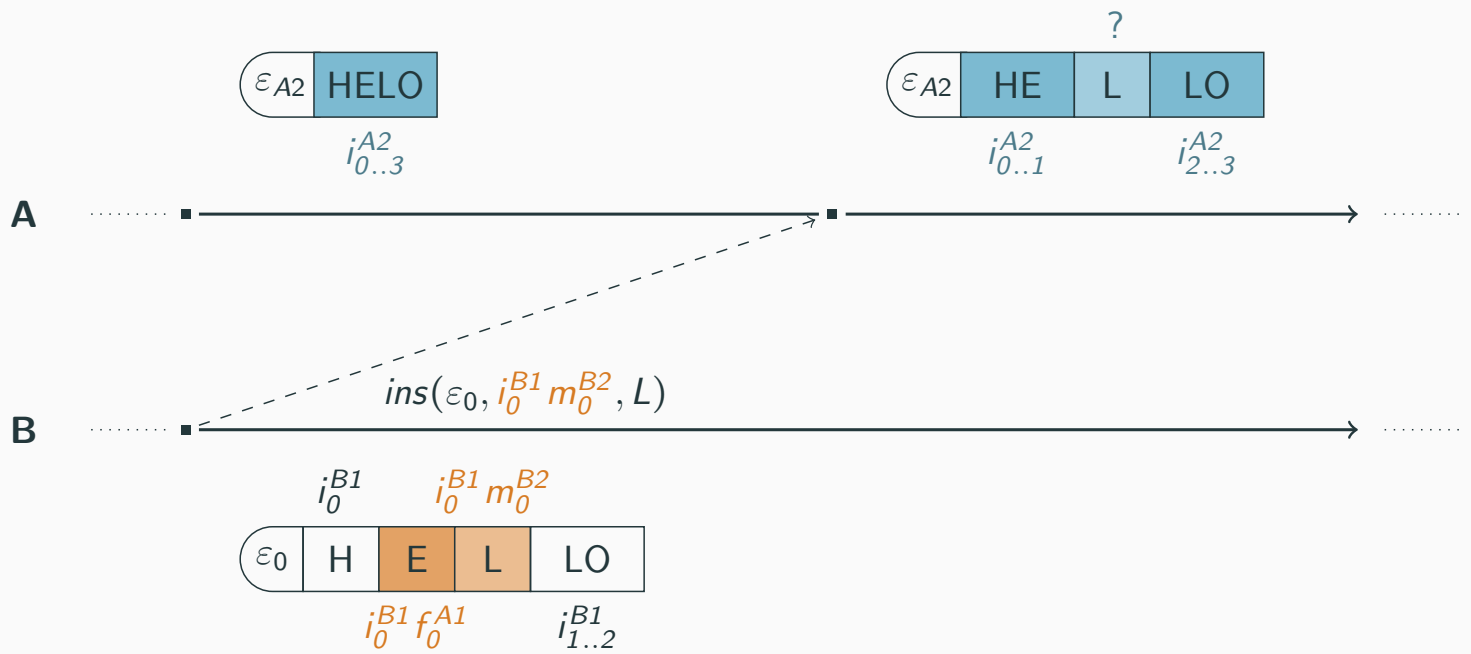


Rappel :

$$renIds_{A2} = [\, i^{B1}_0 \,,\, i^{B1}_0\, f^{A1}_0 \,,\, i^{B1}_1 \,,\, i^{B1}_2 \,]$$

Exemple avec $i^{B1}_0\, m^{B2}_0$

Exemple de `renameId`



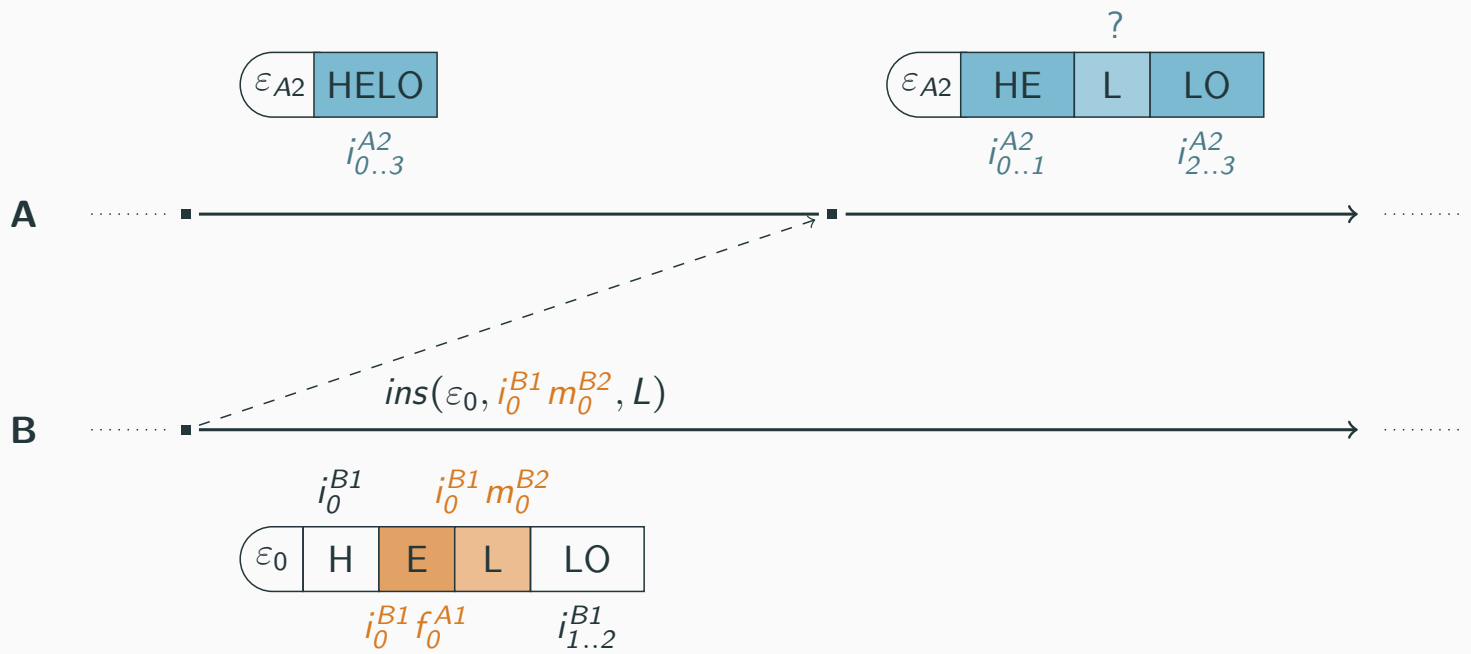
Rappel :

$$renIds_{A2} = [\, i^{B1}_0 \,, \mathbf{i^{B1}_0 f^{A1}_0} \,, i^{B1}_1 \,, i^{B1}_2 \,]$$

Exemple avec $i^{B1}_0 m^{B2}_0$

- Rechercher son prédecesseur dans $renIds_{A2}$: $i^{B1}_0 f^{A1}_0$

Exemple de `renameId`



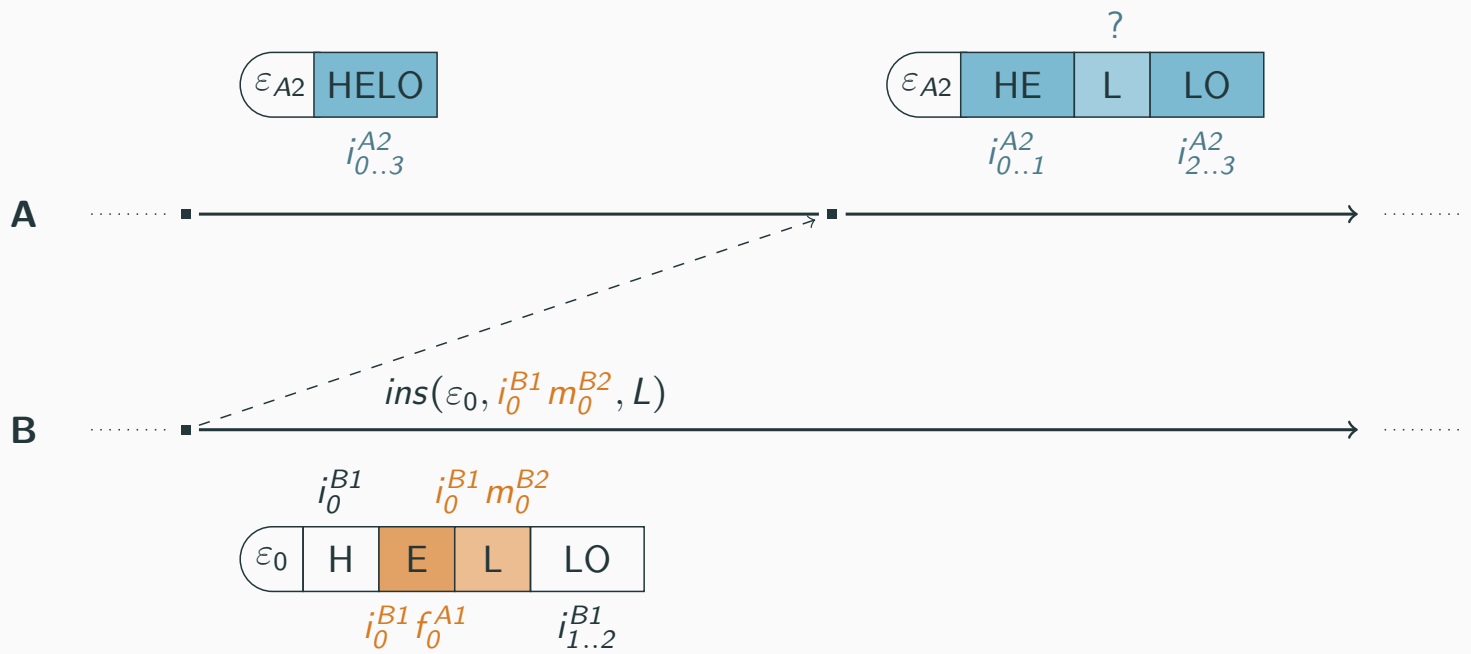
Rappel :

$$renIds_{A2} = [\, i^{B1}_0 \,,\, i^{B1}_0\, f^{A1}_0 \,,\, i^{B1}_1 \,,\, i^{B1}_2 \,]$$

Exemple avec $i^{B1}_0\, m^{B2}_0$

- Rechercher son prédecesseur dans $renIds_{A2}$: $i^{B1}_0\, f^{A1}_0$
- Utiliser son index (1) pour calculer équivalent à époque ε_{A2} : i^{A2}_1

Exemple de `renameId`



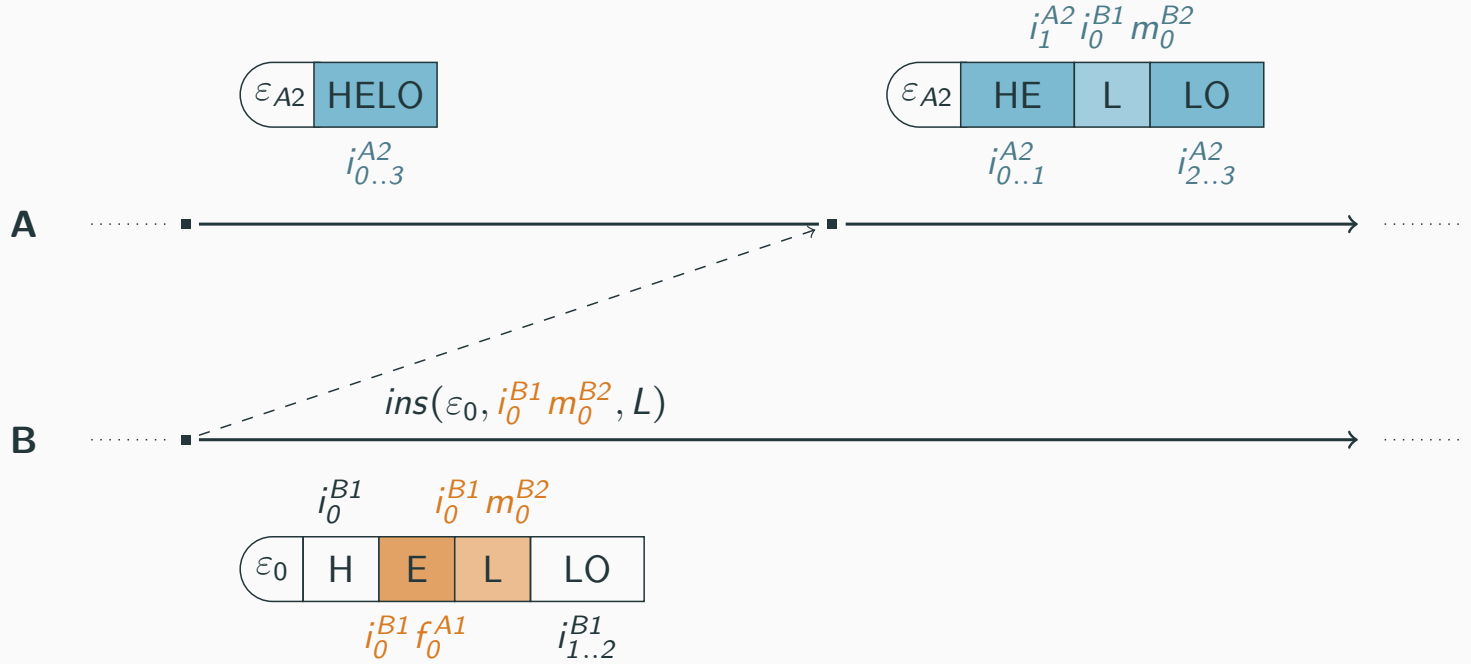
Rappel :

$$renIds_{A2} = [\, i^{B1}_0 \,,\, i^{B1}_0 f^{A1}_0 \,,\, i^{B1}_1 \,,\, i^{B1}_2 \,]$$

Exemple avec $i^{B1}_0 m^{B2}_0$

- Rechercher son prédecesseur dans $renIds_{A2}$: $i^{B1}_0 f^{A1}_0$
- Utiliser son index (1) pour calculer équivalent à époque ε_{A2} : i^{A2}_1
- Préfixer $i^{B1}_0 m^{B2}_0$ par ce dernier : $i^{A2}_1 i^{B1}_0 m^{B2}_0$

Exemple de `renameId`



Rappel :

$$renIds_{A2} = [\, i_0^{B1}\, ,\, i_0^{B1}\, f_0^{A1}\, ,\, i_1^{B1}\, ,\, i_2^{B1}\,]$$

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

- Rechercher son prédecesseur dans $renIds_{A2}$: $i_0^{B1}\, f_0^{A1}$
- Utiliser son index (1) pour calculer équivalent à époque ε_{A2} : i_1^{A2}
- Préfixer $i_0^{B1}\, m_0^{B2}$ par ce dernier : $i_1^{A2}\, i_0^{B1}\, m_0^{B2}$

Objectifs

- Montrer que RenamableLogootSplit satisfait la convergence forte
- Montrer que le mécanisme de renommage améliore les performances de la séquence répliquée (mémoire, calculs, bande-passante)

Objectifs

- Montrer que RenamableLogootSplit satisfait la convergence forte
- Montrer que le mécanisme de renommage améliore les performances de la séquence répliquée (mémoire, calculs, bande-passante)

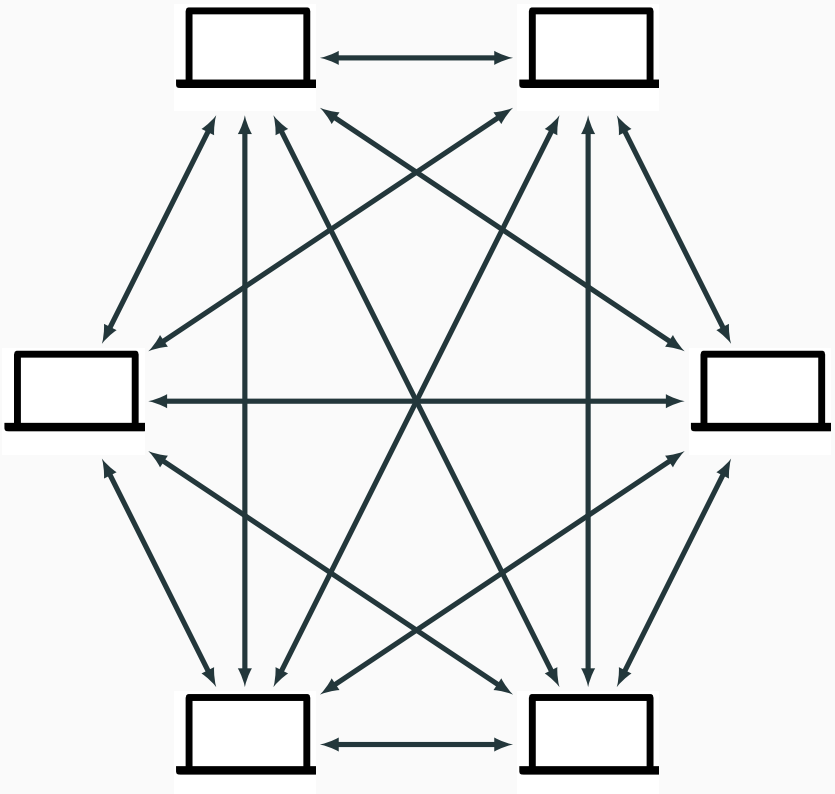
Conduite d'une évaluation expérimentale

Absence d'un jeu de données de sessions d'édition collaborative

Absence d'un jeu de données de sessions d'édition collaborative

Mise en place de simulations pour générer un jeu de données

Simulations - Architecture



- 10 noeuds éditent collaborativement un document
- Topologie réseau entièrement maillée
- Ne considère pas de pannes ou de pertes de message

Simulations - Modifications

Noeuds utilisent LogootSplit (LS) ou RenamableLogootSplit (RLS)

Simulations - Modifications

Noeuds utilisent LogootSplit (LS) ou RenamableLogootSplit (RLS)

Se décompose en 2 phases

1. Génération du contenu (80% d'*insert*, 20% de *remove*)
2. Édition (50/50%)

Noeuds passent à la phase 2 quand leur copie locale atteint une taille donnée (15 pages - 60k caractères)

Simulations - Modifications

Noeuds utilisent LogootSplit (LS) ou RenamableLogootSplit (RLS)

Se décompose en 2 phases

1. Génération du contenu (80% d'*insert*, 20% de *remove*)
2. Édition (50/50%)

Noeuds passent à la phase 2 quand leur copie locale atteint une taille donnée (15 pages - 60k caractères)

Nombre d'opérations : 15k par noeud, 150k au total

Simulations - Mécanisme de renommage

Noeuds de renommage

- 1 à 4 noeuds effectuent une opération *rename* toutes les 30k opérations
- Opérations *rename* générées à un point donné sont concurrentes

Simulations - Sorties

- Instantané de l'état de chaque noeud à différents points de la simulation (10k opérations et état final)
- Journal des opérations de chaque noeud

*. Code des simulations et benchmarks : `https://github.com/coast-team/mute-bot-random`

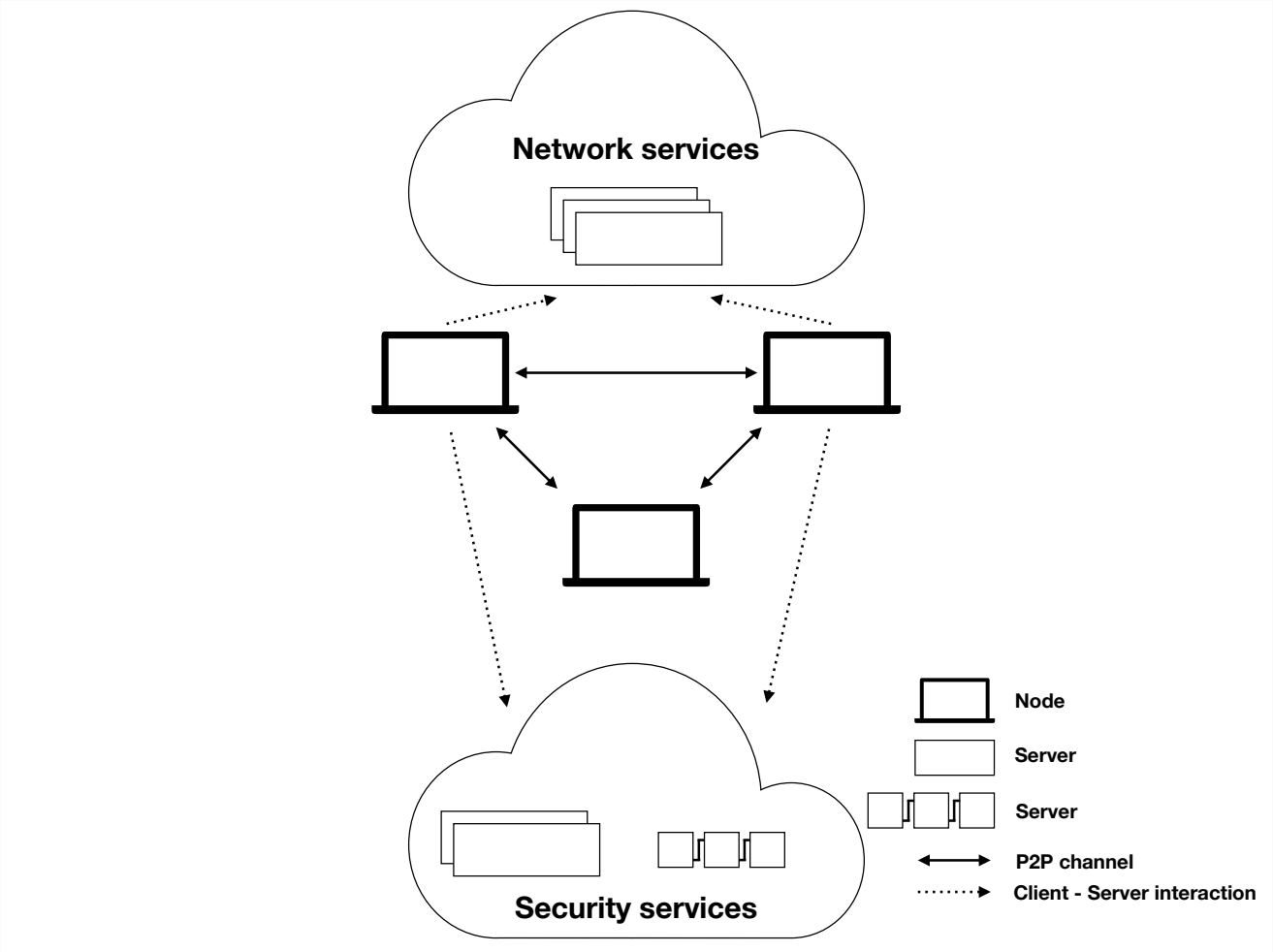
Simulations - Sorties

- Instantané de l'état de chaque noeud à différents points de la simulation (10k opérations et état final)
- Journal des opérations de chaque noeud

Permet de conduire évaluations sur ces données*

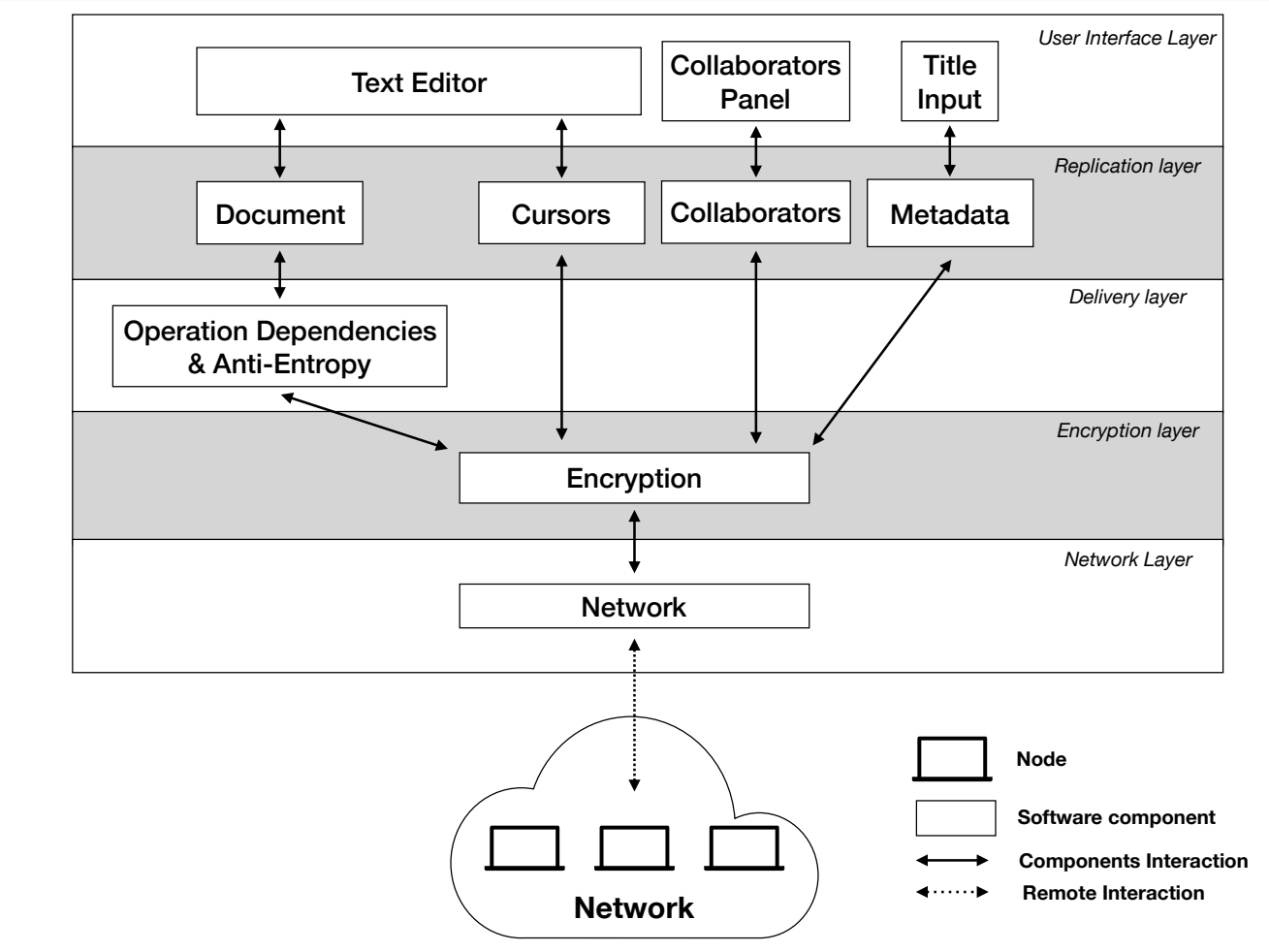
*. Code des simulations et benchmarks :
`https://github.com/coast-team/mute-bot-random`

Architecture système de MUTE



- Toto

Architecture logicielle de MUTE



- Toto