

- Retourne  $S''$ , la Least Upper Bound (LUB) de  $S$  et  $S'$  (c.-à-d.  $\#S''' \cdot \text{merge}(S, S') < S''' < S''$ )

Et bénéficie de son principal avantage : synchronisation possible entre deux pairs en fusionnant leur états, peu importe le nombre de modifications les séparant.

- Spécificité des Delta-based CRDTs est de proposer une synchronisation par différence d'états. Plutôt que de diffuser l'entièreté de l'état pour permettre aux autres pairs de se mettre à jour, idée est de seulement transmettre la partie de l'état ayant été mise à jour. Correspond à un élément irréductible du sup-demi-treillis. Permet ainsi de mettre en place une synchronisation en temps réel de manière efficace. Et d'utiliser la synchronisation par fusion d'états complets pour compenser les défaillances du réseau
- Ainsi, ce nouveau type de CRDTs semble allier le meilleur des deux mondes :
  - Absence de contrainte sur le réseau autre que la livraison à terme
  - Propagation possible en temps réel des modifications

Semble donc être une solution universelle :

- Utilisable peu importe la fiabilité réseau à disposition
- Empreinte réseau du même ordre de grandeur qu'un Op-based CRDT
- Utilisable peu importe la fréquence de synchronisation désirée

Pose la question de l'intérêt des autres types de CRDTs.

- Delta-based CRDT est un State-based CRDT dont on a identifié les éléments irréductibles et qui utilise ces derniers pour la propagation des modifications plutôt que l'état complet. Famille des State-based CRDTs semble donc rendue obsolète par celle des Delta-based CRDTs. À confirmer.
- Les Op-based CRDTs proposent une spécification différente du type répliqué de leur équivalent Delta-based, généralement plus simple. À première vue, famille des Op-based CRDTs semble donc avoir la simplicité comme avantage par rapport à celle des Delta-based CRDTs. S'agit d'un paramètre difficilement mesurable et auquel on peut objecter si on considère qu'un Op-based CRDT s'accompagne d'une couche livraison de messages, qui cache sa part de complexité. Intéressant d'étudier si la spécification différente des Op-based CRDTs présente d'autres avantages par rapport aux Delta-based CRDTs : performances (temps d'intégration des modifications, délai de convergence...), fonctionnalités spécifiques (composition, undo...)
- But serait de fournir des guidelines sur la famille de CRDT à adopter en fonction du cas d'utilisation.

#### 1.2.4 Approfondissement du patron de conception de Pure Op-based CRDTs

- BAQUERO et al. [12] proposent un framework pour concevoir des CRDTs synchronisés par opérations : Pure Op-based CRDTs. Ce framework a plusieurs objectifs :

- (i) Proposer une approche partiellement générique pour définir un CRDT synchronisé par opérations.
  - (ii) Factoriser les métadonnées utilisées par le CRDT pour le mécanisme de résolution de conflits, notamment pour identifier les éléments, et celles utilisées par la couche livraison, notamment pour identifier les opérations.
  - (iii) Inclure des mécanismes de GC de ces métadonnées pour réduire la taille de l'état.
- Pour cela, les auteurs se limitent aux CRDTs purs synchronisés par opérations, c.-à-d. les CRDTs dont les modifications enrichies de leurs arguments et d'une estampille fournie par la couche de livraison des messages sont commutatives. Pour ces CRDTs, les auteurs proposent un framework générique permettant leur spécification sous la forme d'un PO-Log associé à une couche de livraison Reliable Causal Broadcast (RCB).
  - Les auteurs définissent ensuite le concept de stabilité causale. Ce concept leur permet de retirer les métadonnées de causalité des opérations du PO-Log lorsque celles-ci sont déterminées comme étant causalement stables.
  - Finalement, les auteurs définissent un ensemble de relations, spécifiques à chaque CRDT, qui permettent d'exprimer la *redondance causale*. La redondance causale permet de spécifier quand retirer une opération du PO-Log, car rendue obsolète par une autre opération.
  - Cette approche souffre toutefois de plusieurs limites. Tout d'abord, elle repose sur l'utilisation d'une couche de livraison RCB. Cette couche satisfait le modèle de livraison causale. Mais pour rappel, ce modèle induit l'ajout de données de causalité précises à chaque opération, sous la forme d'un vecteur de version ou d'une barrière causale. Nous jugeons ce modèle trop coûteux pour un système P2P dynamique à large échelle.
  - En plus du coût induit en termes de métadonnées et de bande-passante, le modèle de livraison causale peut aussi introduire un délai superflu dans la livraison des opérations. En effet, ce modèle impose que tous les messages précédant un nouveau message d'après la relation *happens-before* soient eux-mêmes livrés avant de livrer ce dernier. Il en résulte que des opérations peuvent être mises en attente par la couche livraison, e.g. suite à la perte d'une de leurs dépendances d'après la relation *happens-before*, alors que leurs dépendances réelles ont déjà été livrées et que les opérations sont de fait intégrables en l'état. Plusieurs travaux [13, 14] ont noté ce problème. Pour y répondre et ainsi améliorer la réactivité du framework Pure Op-based, ils proposent d'exposer les opérations mises en attente par la couche livraison au CRDT. Bien que fonctionnelle, cette approche induit toujours le coût d'une couche de livraison respectant le modèle de livraison causale et nous fait considérer la raison de ce coût, le modèle de livraison n'étant alors plus respecté.
  - Ensuite, ce framework impose que la modification **prepare** ne puisse pas inspecter l'état courant du noeud. Cette contrainte est compatible avec les CRDTs pour les types de données simples qui sont considérés dans [12], e.g. le Compteur ou l'Ensemble. Elle empêche cependant l'expression de CRDTs pour des types de données

plus complexes, e.g. la Séquence ou le Graphe. *Matthieu: TODO : À confirmer pour le graphe* Nous jugeons dommageable qu'un framework pour la conception de CRDTs limite de la sorte son champ d'application.

- Finalement, les auteurs ne considèrent que des types de données avec des modifications à granularité fixe. Ainsi, ils définissent la notion de redondance causale en se limitant à ce type de modifications. Par exemple, ils définissent que la suppression d'un élément d'un ensemble rend obsolète les ajouts précédents de cet élément. Cependant, dans le cadre d'autres types de données, e.g. la Séquence, une modification peut concerner un ensemble d'éléments de taille variable. Une opération peut donc être rendue obsolète non pas par une opération, mais par un exemple d'opérations. Par exemple, les suppressions d'éléments formant une sous-chaîne rendent obsolète l'insertion de cette sous-chaîne. Ainsi, la notion de redondance causale est incomplète et souffre de l'absence d'une notion d'obsolescence partielle d'une opération.
- Pour répondre aux différents problèmes soulevés, nous souhaitons proposer une extension du framework Pure Op-based CRDTs. Nos objectifs sont les suivants :
  - (i) Proposer un framework mettant en lumière la présence et le rôle de deux modèles de livraison :
    - (i) Le modèle de livraison minimale requis par le CRDT pour assurer la convergence forte à terme.
    - (ii) Le modèle de livraison utilisé par le système, qui doit être égal ou plus contraint que modèle de livraison minimal du CRDT. Ce second modèle de livraison est une stratégie permettant au système de respecter un modèle de cohérence donné et régissant les règles de compaction de l'état. Il peut être amené à évoluer en fonction de l'état du système et de ses besoins. Par exemple, un système peut par défaut utiliser le modèle de livraison causale pour assurer le modèle de cohérence causal. Puis, lorsque le nombre de noeuds atteint un seuil donné, le système peut passer au modèle de livraison FIFO pour assurer le modèle de cohérence PRAM afin de réduire les coûts en bande-passante.
  - (ii) Rendre accessible la notion de redondance causale à la couche de livraison, pour détecter au plus tôt les opérations désormais obsolètes et prévenir leur diffusion.
  - (iii) Identifier et classifier les mécanismes de résolution de conflits, pour déterminer lesquels sont indépendants de l'état courant pour la génération des opérations et lesquels nécessitent d'inspecter l'état courant dans **prepare**.