# Cohérence des données dans les environnements d'édition collaborative

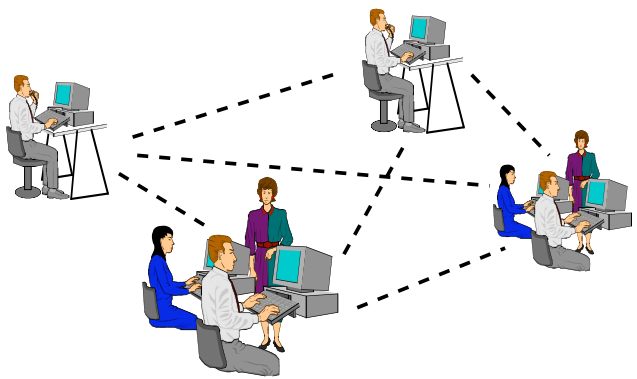
Pascal Molli

Nancy-Université

Habilitation à diriger des recherches
26 avril 2007

# Introduction

- Édition Collaborative : Produire des documents à plusieurs en étant distribué dans le temps, l'espace et à travers les organisations.
- Production de code avec des équipes virtuelles (SourceForge), production de pages wiki (Wikipedia), écriture de manuels...



- Fournir des modèles et des outils pour rendre la coopération efficace...

# Introduction

**Partager les données** de FTP au Gestionnaire de configuration...

**Communiquer** Mail, Messagerie instantanée, visio-conférence, partage d'application...

**Se Coordonner** Procédés, de la TodoList au Workflow,

**Conscience de groupe** Qui fait quoi ? quand ? comment ? pourquoi ? qu'ont-ils fait pendant mon absence ? Que vont ils faire?

# Problèmes scientifiques

- Gestion des données partagées dans un environnement collaboratif:
  - Données critiques en accès Write/Write...
  - Risque de corruption des données, risque de pertes de mise-à-jours
  - Risque de travail sur données obsolètes
  - Risque de modifications en aveugle (travail concurrent inutile)...
- 1 système collaboratif = 1 système distribué avec accès concurrents
- Cohérence des données : Quels critères de cohérence ? Quels protocoles ? Comment gérer le facteur humain ?

# Plan de la présentation

# État de l'art

## Éviter les conflits...

- Verrouillage partiel ou complet...
- travail sur données disjointes...

- Mail + Données disjointes...
- Turn taking:
  - Netmeeting en synchrone. Partage d'application et passage du jeton entre les utilisateurs connectés. Pas de support asynchrone...
  - Liste de diffusion en asynchrone. Chaque personne modifie à son tour le document et passe au prochain destinataire. Pas de parallélisation...
- Verrouillage. L'utilisateur ou le système verrouille la zone qu'il est en train d'éditer. Pas de modification concurrente sur cette zone. Période de verrouillage non prévisible...

# Approche réplication optimiste

## Convergence...

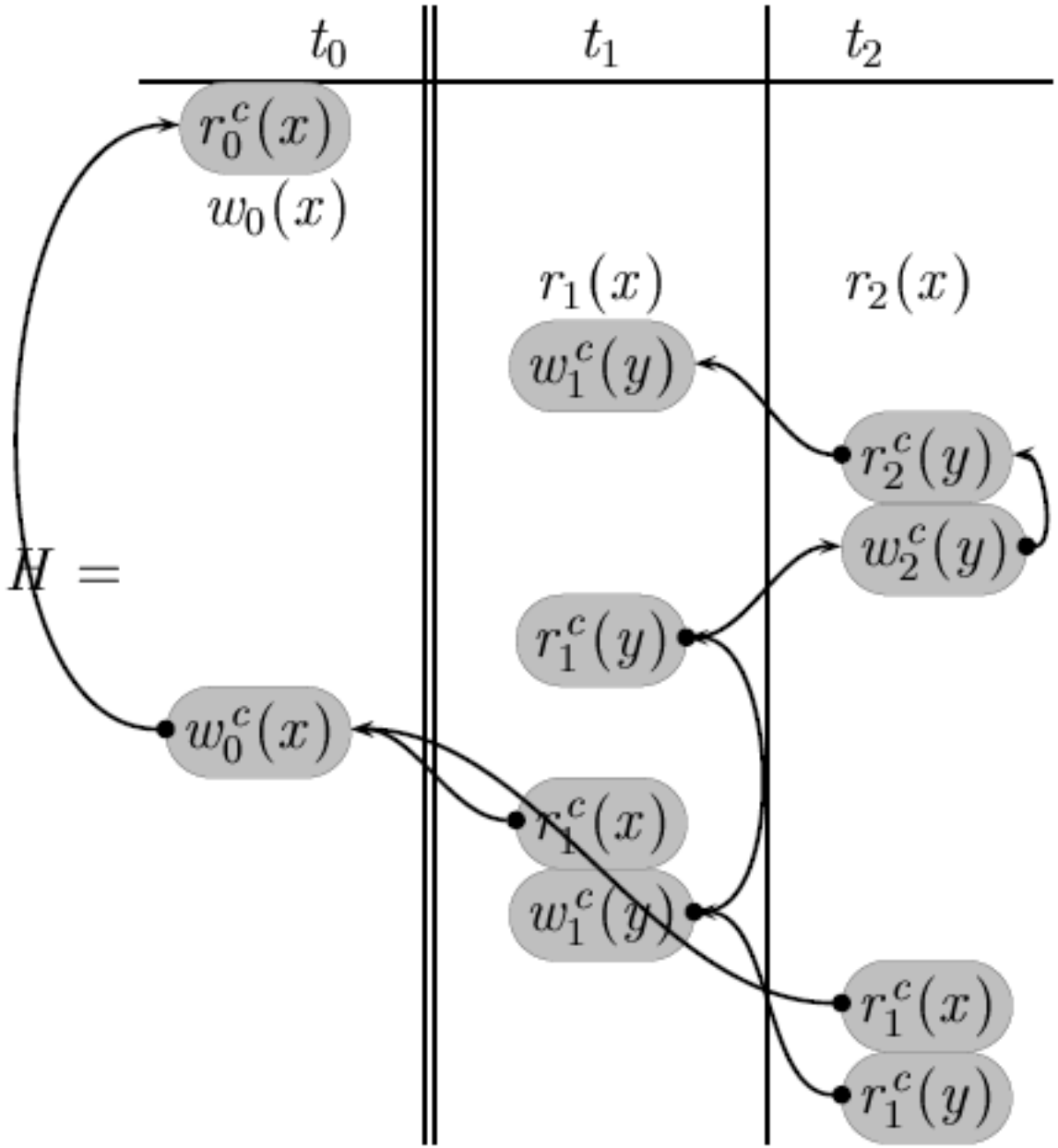
- Chaque intervenant travaille sur une copie...
- Quand le système est au repos toutes les copies sont identiques...

- Multi-version, fusion d'état... Copier-modifier-fusionner... (SCM), Toutes les 10s avec Google doc...
- Sérialisation et gestion de conflits... Undo/do/redo (Bayou)...
- Approche base de données multi-maître ou transactions longues
- Résolution de contraintes (IceCube)...
- Transformées opérationnelles...

# Plan de la présentation

# Coo-Sérialisabilité

- Les utilisateurs accèdent aux données à travers des transactions...
- Étendre la sérialisabilité pour supporter des exécutions non-sérialisable...
- Certaines opérations Read/Write ne servent qu'à coopérer et ne contribuent pas au résultat final $\rightarrow$ opérations ignorées
- Si Les opérations restantes sont sérialisables : OK
- Sinon, on groupe les transactions qui doivent converger vers un résultat commun...


$t_0$
$t_1$
$t_2$
$r_0^c(x)$
$w_0(x)$
$r_1(x)$
$r_2(x)$
$w_1^c(y)$
$r_2^c(y)$
$w_2^c(y)$
$H =$
$r_1^c(y)$
$w_0^c(x)$
$r_1^c(x)$
$w_1^c(y)$
$r_1^c(x)$
$r_1^c(y)$

# Limitations

- Dans le cas coopératif: Oblige les transactions à converger pour terminer... Finalement Sérialisabilité $\rightarrow$ Convergence
- Changer de monde : Réplication optimiste et convergence...
- Choix des transformées opérationnelles :
  - Indépendant du type des données (vs approche Bd Multi-master),
  - Gestion de la cohérence sans site central (vs Bayou ou IceCube),
  - Cadre théorique fort (vs Gestionnaire de conf).

# Plan de la présentation

# Transformées opérationnelles (OT)

- Modèle de réplication optimiste issu des éditeurs temps-réel
- considère $n$ sites, chacun avec une réplique...
- une opération est :
  1. exécutée localement,
  2. diffusée aux autres sites,
  3. reçue sur un site,
  4. *transformée par rapport aux op concurrentes*,
  5. re-exécutée.

## 2 composants :

- un algorithme d'intégration : diffusion, intégration générique : GOT, GOTO, SOCT2,3,4,5, Adopted, COT
- des fonctions de transformation dépendant des types données.

## Correction

- Les algorithmes garantissent causalité et convergence si les fonctions de transformation vérifient au moins $C_1$ pour SOCT4, COT et GOT:

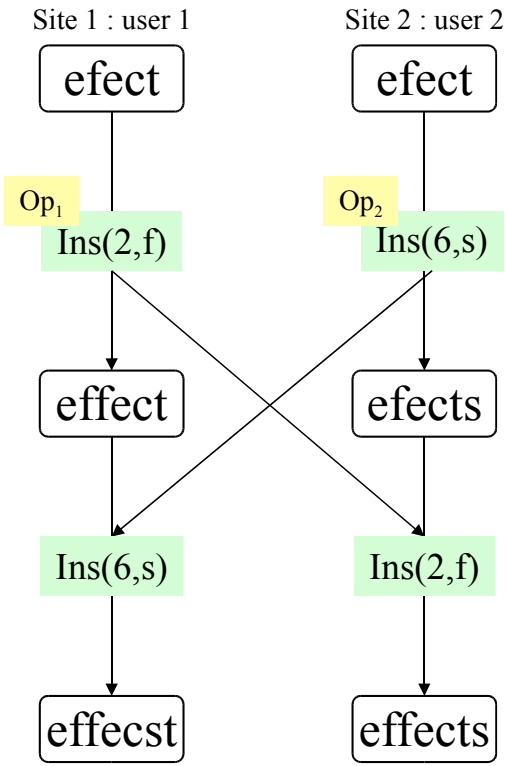
$$C_1 : op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$$

- et aussi $C_2$ pour GOTO, SOCT2,3, aDopted...:

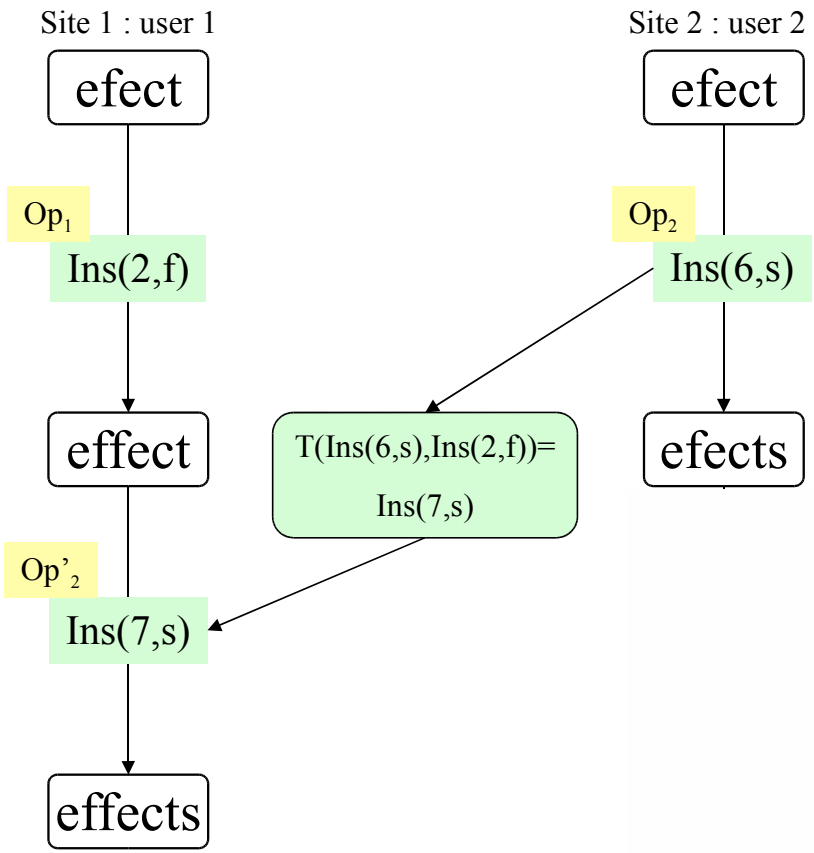
$$C_2 : T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2))$$

# Problématique : Convergence

- Intégration incorrecte d'opérations concurrentes...
- *Lorsque le système est stable, les répliques doivent converger*



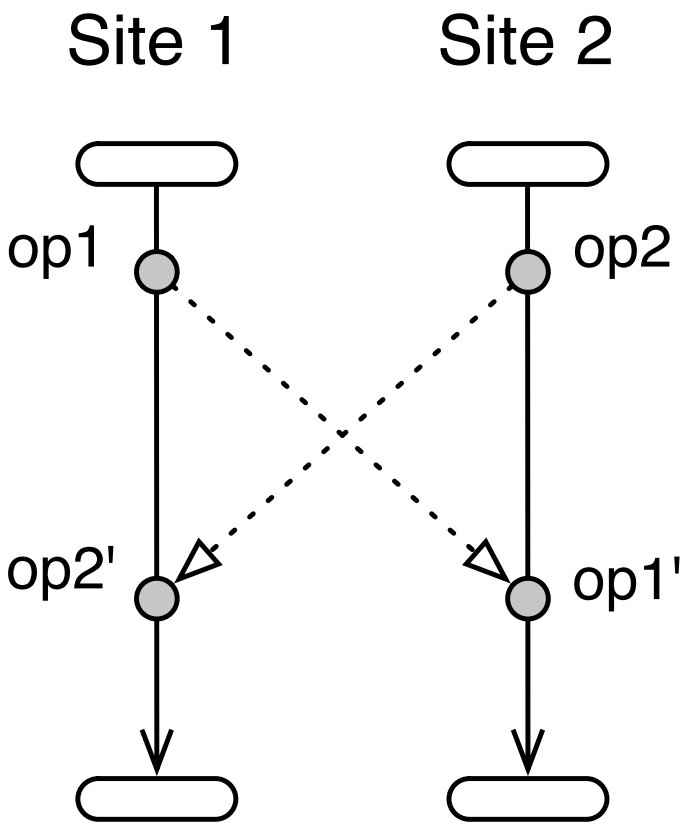
# Transformer les opérations...




$$
\begin{array}{l}
T(Ins(p_1, c_1), Ins(p_2, c_2)) :- \\
\quad \textbf{if} \ \ (p_1 < p_2) \\
\quad\quad \textbf{return} \ Ins(p_1, c_1) \\
\quad \textbf{else} \\
\quad\quad \textbf{return} \ Ins(p_1 + 1, c_1) \\
\quad \textbf{endif}
\end{array}
$$

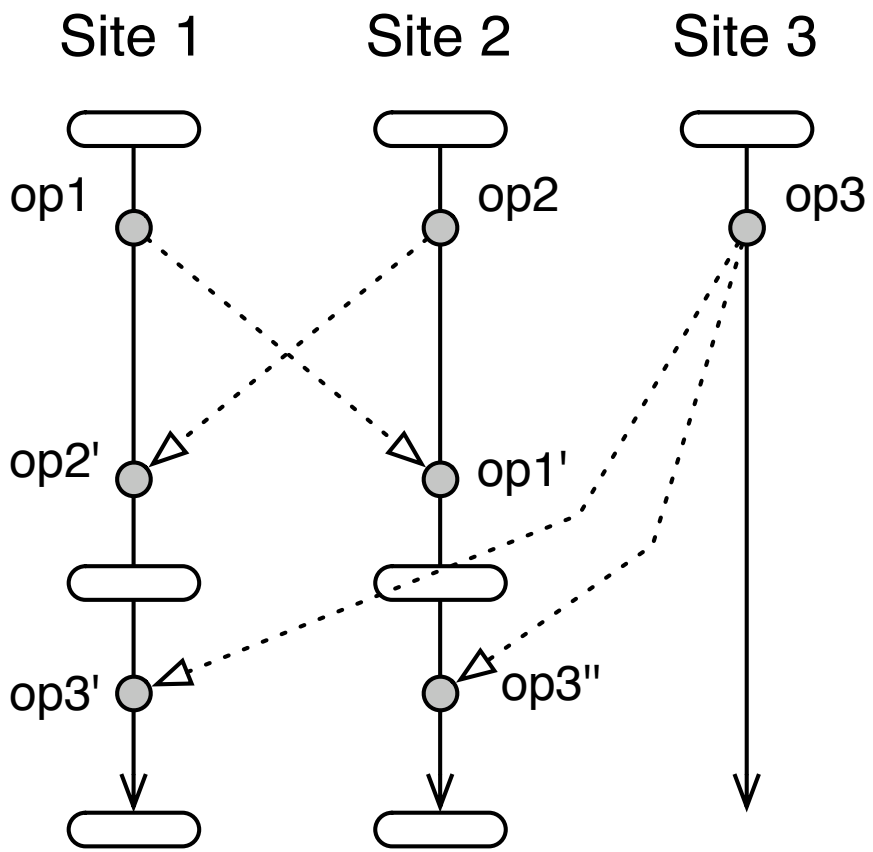
- T($op_2$:opération, $op_1$:opération) = $op'_2$
  - $op_1$ et $op_2$ concurrentes, définies sur le même état $S$
  - $op'_2$ même effets que $op_2$, mais sur l'état $S.op_1$

# OT : Convergence - condition $C_1$




$$op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$$

# OT : Convergence - condition $C_2$




$$T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2))$$

# $C_1$ et/ou $C_2$

- SOCT4 et COT ne nécessite que $C_1$ mais force l'utilisation d'un ordre total continu: site central ou algorithmes de consensus
- GOT nécessite un ordre total discontinu mais utilise la stratégie du undo/do/redo. Pas performant (et peut-être pas juste aussi...)
- GOTO, SOCT2,3, Adopted nécessite $C_2$ mais ne requiert aucun ordre spécial. Pas de sites centraux, pas de consensus...

# OT : Problèmes

- Conception et vérification des fonctions de transformation
- écriture de T pour chaque couple d'opérations $T(ins, ins)$, $T(ins, del)$, $T(del, ins)$, $T(del, del)$
- vérification des conditions $C_1$ et $C_2$
  - combinatoire ($> 100$ cas pour une chaîne de caractères),
  - processus itératif,
  - tâche répétitive et laborieuse

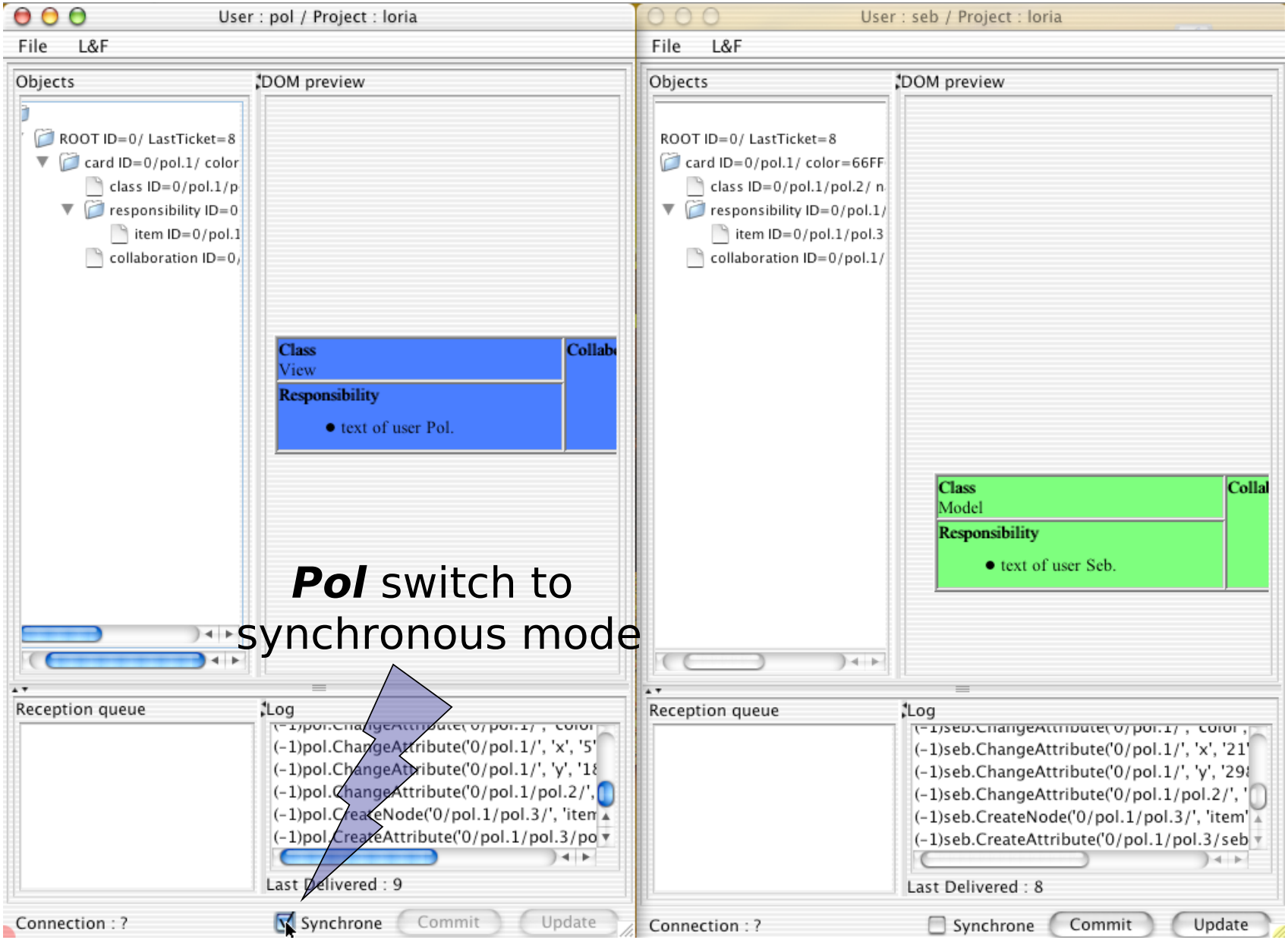
# Objectifs

- Utiliser OT pour faire autre chose que des éditeurs temps réel
- Faire des éditeurs multi-synchrones...
- Faire des gestionnaires de configuration...
- Faire des synchroniseurs...

# Plan de la présentation

# SAMS : Éditeur Synchrone, Asynchrone, Multi-Synchrone [2001]

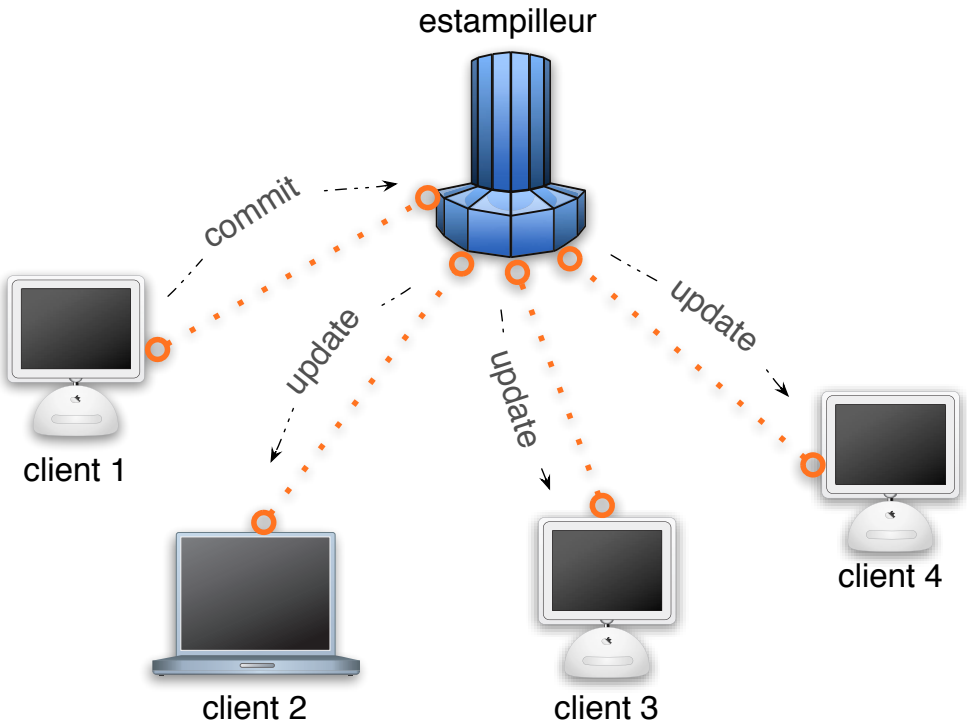
- Basé sur SOCT4+Transformées vérifiant $C_1$.
- SAMS : Utiliser le meilleurs mode de travail au meilleur moment pour tout ou partie du groupe...
- Pouvoir opter à tout moment pour un mode de travail synchrone ou asynchrone...
- Toujours une bonne idée aujourd'hui... : Google docs avec support deconnecté (FireFox3) = éditeur SAMS


User : pol / Project : loria
File L&F
Objects
DOM preview
ROOT ID=0/ LastTicket=8
card ID=0/pol.1/ color
class ID=0/pol.1/p
responsibility ID=0
item ID=0/pol.1
collaboration ID=0/
Class
View
Responsibility
text of user Pol.
Reception queue
Log
(-1)pol.ChangeAttribute('0/pol.1/', 'x', '5'
(-1)pol.ChangeAttribute('0/pol.1/', 'y', '1
(-1)pol.ChangeAttribute('0/pol.1/pol.2/',
(-1)pol.CreateNode('0/pol.1/pol.3/', 'item
(-1)pol.CreateAttribute('0/pol.1/pol.3/po
Last Delivered : 9
Connection : ?
Synchrone
Commit
Update
User : seb / Project : loria
File L&F
Objects
DOM preview
ROOT ID=0/ LastTicket=8
card ID=0/pol.1/ color=66FF
class ID=0/pol.1/pol.2/ n
responsibility ID=0/pol.1/
item ID=0/pol.1/pol.3
collaboration ID=0/pol.1/
Class
Model
Responsibility
text of user Seb.
Reception queue
Log
(-1)seb.ChangeAttribute('0/pol.1/', 'x', '21'
(-1)seb.ChangeAttribute('0/pol.1/', 'y', '29
(-1)seb.ChangeAttribute('0/pol.1/pol.2/',
(-1)seb.CreateNode('0/pol.1/pol.3/', 'item'
(-1)seb.CreateAttribute('0/pol.1/pol.3/seb
Last Delivered : 8
Connection : ?
Synchrone
Commit
Update
Pol switch to synchronous mode

# Plan de la présentation

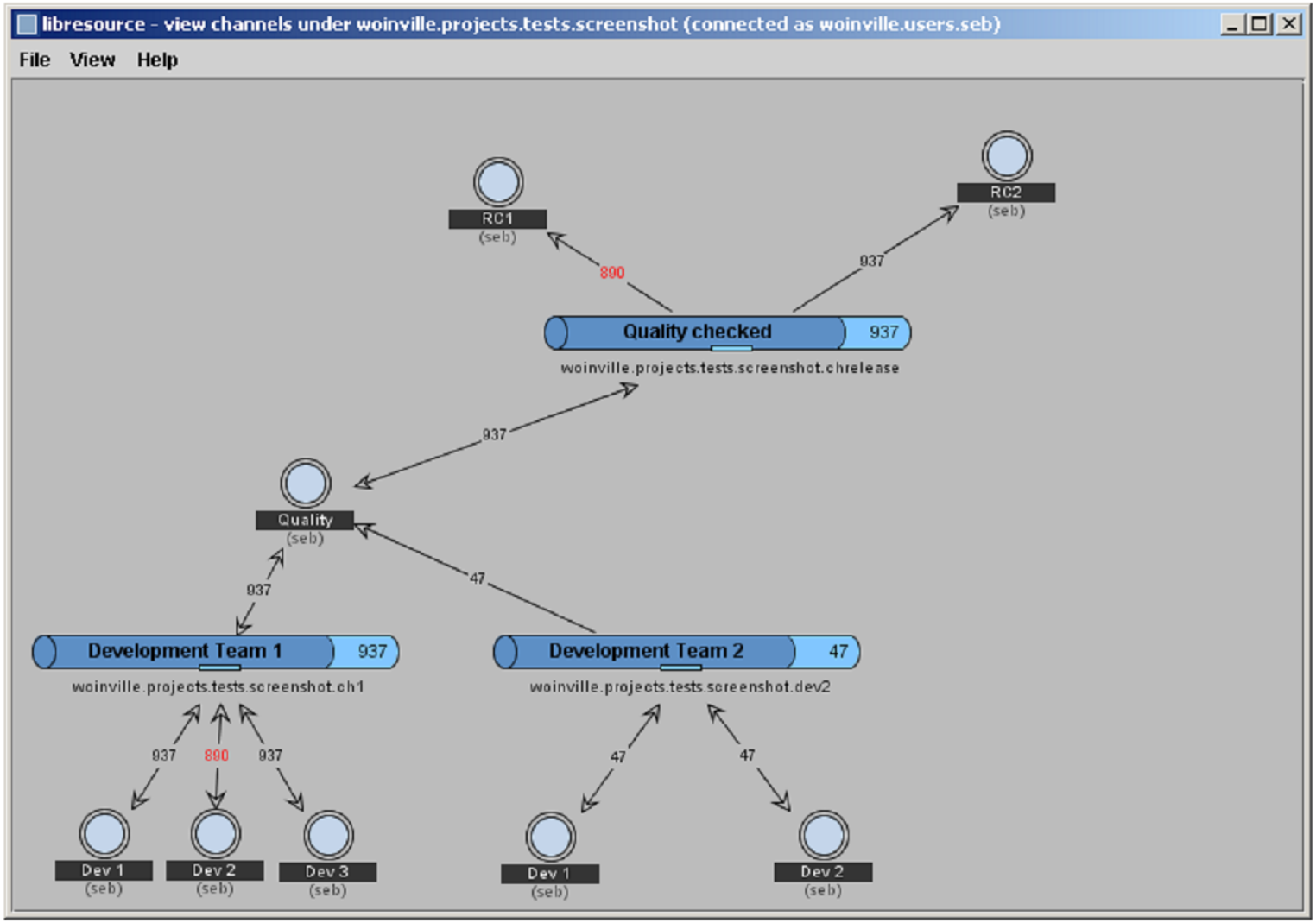
# SO6 : un gestionnaire de configuration

- Un outil de gestion de configuration reposant OT
  - Centralisé car $C_1$+ordre total [Vidot00]




  - Des fonctions de transformations pour :
    - Système de fichiers, texte et XML.
  - Extensible à de nouveaux types de données
- Distribué commercialement et de façon libre

# So6 et Dataflow



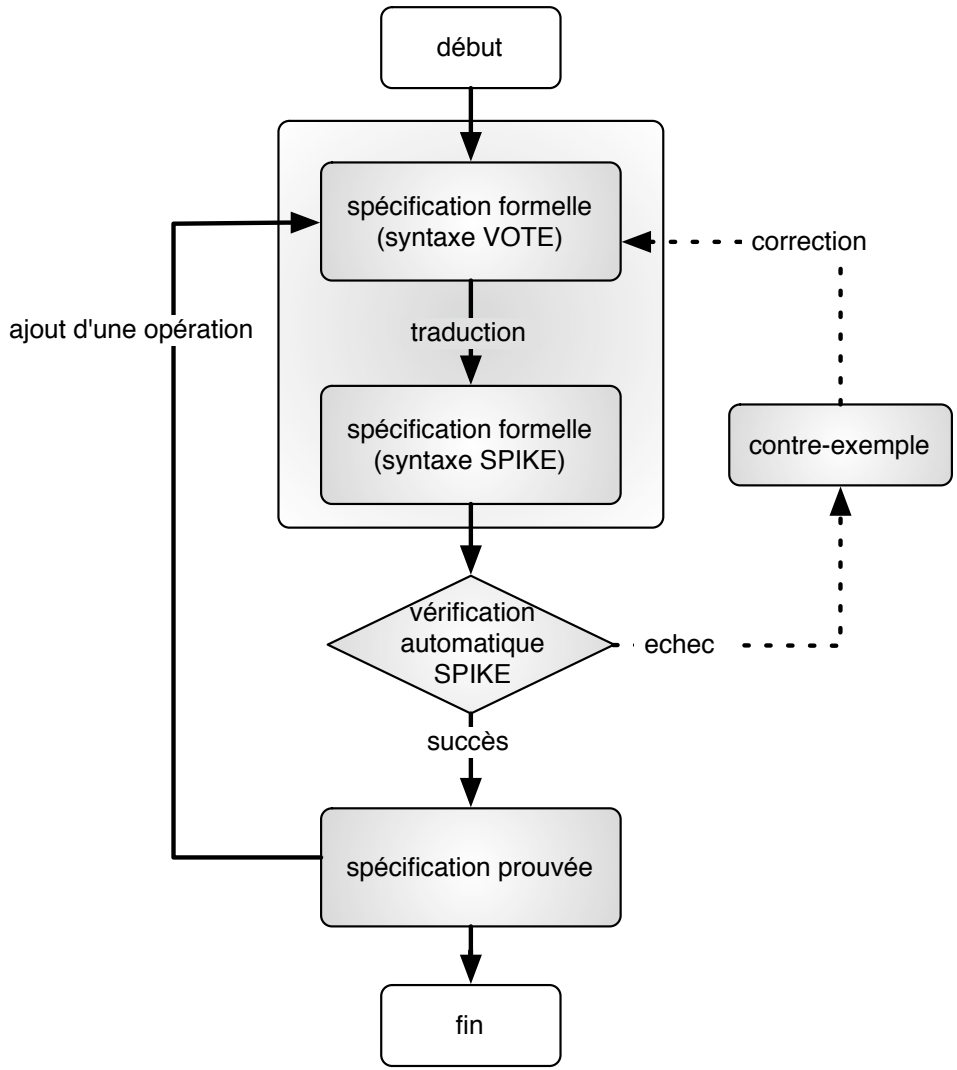
# Limitations So6

- Topologie en arbre...
- Sinon, la convergence peut être violée. L'ordre n'est plus total continu.
- Pour n'importe quelle topologie : $C_2$, Mais problèmes de preuves.

# Plan de la présentation

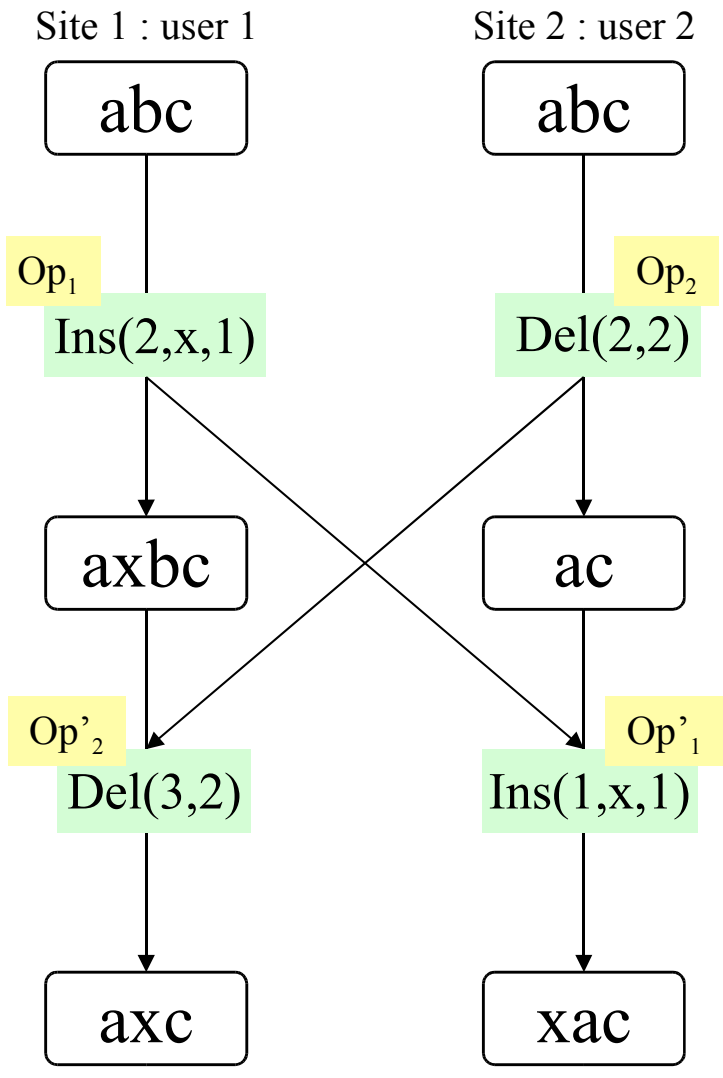
# VOTE : Approche

- Vérification formelle et automatique
  - $\Longrightarrow$ sans erreur de calcul
  - $\Longrightarrow$ plus rapide
- Entrée :
  - Conditions $C_1$ et $C_2$,
  - Fonctions T
- Sortie :
  - OK,
  - KO + contre-exemple



# VOTE : Contre-exemple [Ellis'89]

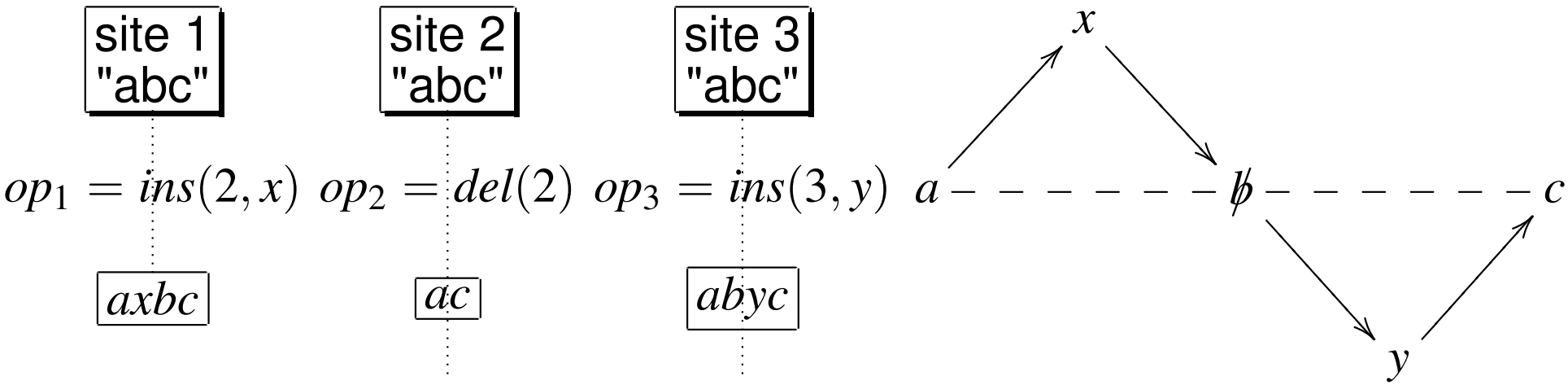
```
Operations :
 op1 = Ins(p1,c1,pr1)
 op2 = Del(p2,pr2)

Conditions :
      p1 < length(u5)
 and p2 <= length(u5)
 and p1 = p2
 and pr1 <> pr2
```



# VOTE : Résultats

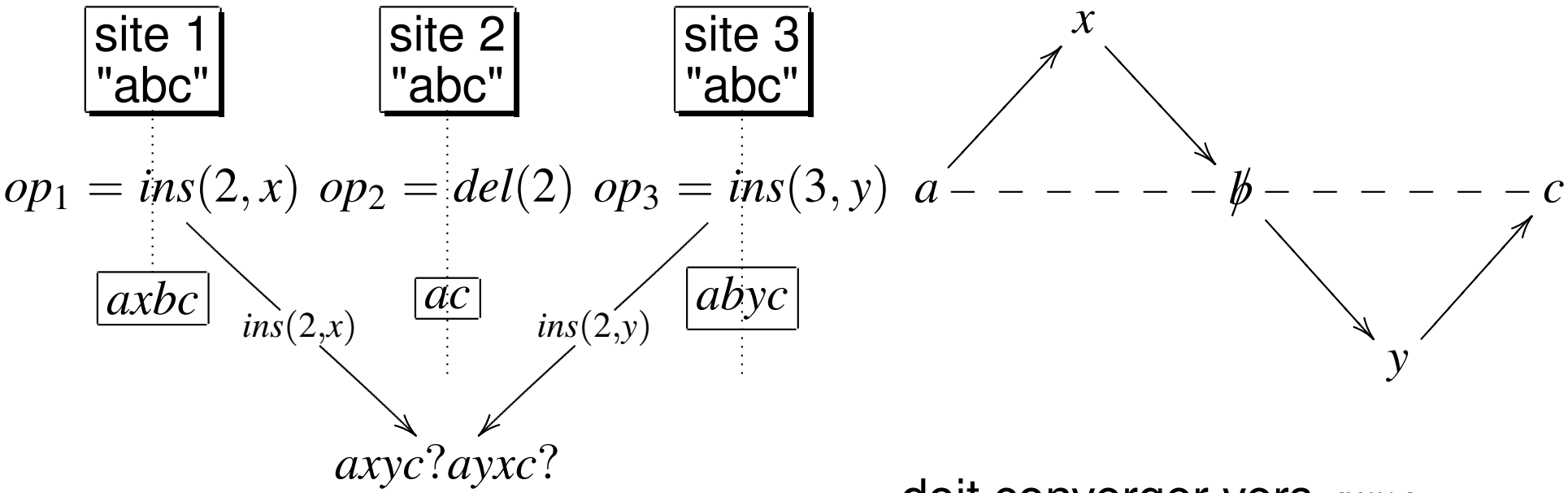
- Une approche pour concevoir et vérifier T :
  - Retrouver les contre-exemples pour [Ellis'89], [Ressel'96], [Sun'98]
  - De nouvelles fonctions vérifiant $C_1$ (système de fichiers, fichier texte, XML)
  - De nouveaux contre-exemples pour [Suleiman'98], [Sun'00], [Imine'03], [Imine'05]
- Aucune fonction ne satisfait $C_1$ et $C_2$

# VOTE : Problème récurrent




doit converger vers $axyc$

# VOTE : Problème récurrent



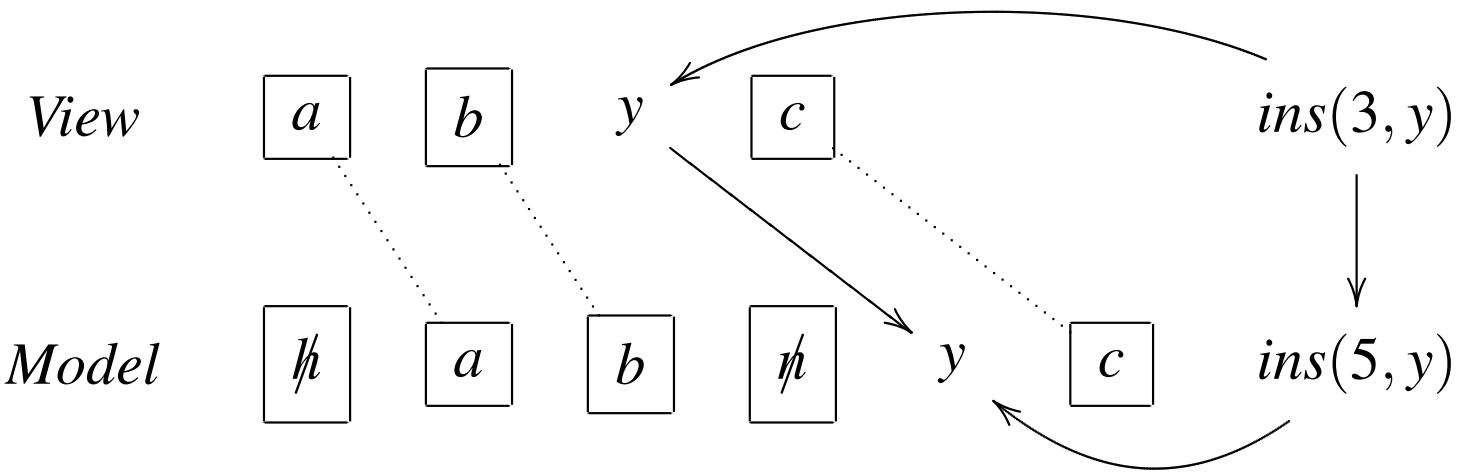

doit converger vers $axyc$

- positionnement absolu est sujet à des pertes d'informations
- différentes propositions OT cherchent à retrouver ces informations

# Plan de la présentation

# Tombstone Transformation Functions

- Idée: ne pas détruire...
- Différencier modèle et vue...



# TTF

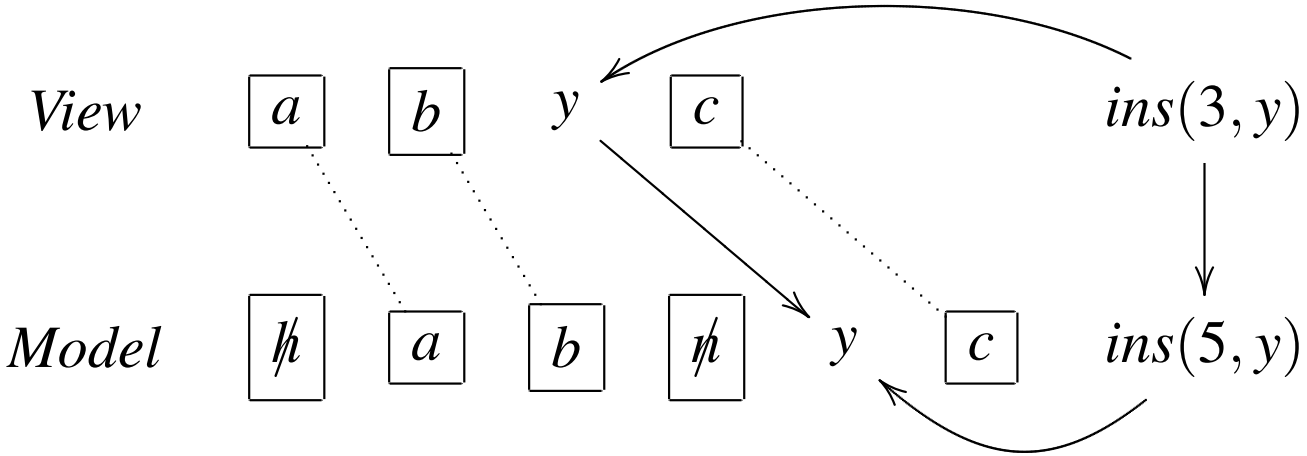
$T(ins(p_1, c_1, sid_1), ins(p_2, c_2, sid_2))$ :−
**if** $(p_1 < p_2)$ **return** $ins(p_1, c_1, sid_1)$
**else if** $(p_1 = p_2$ **and** $sid_1 < sid_2)$ **return** $ins(p_1, c_1, sid_1)$
**else return** $ins(p_1 + 1, c_1, sid_1)$

$T(del(p_1, sid_1), ins(p_2, c_2, sid_2))$ :−
**if** $(p_1 < p_2)$ **return** $del(p_1, sid_1)$
**else return** $del(p_1 + 1, sid_1)$

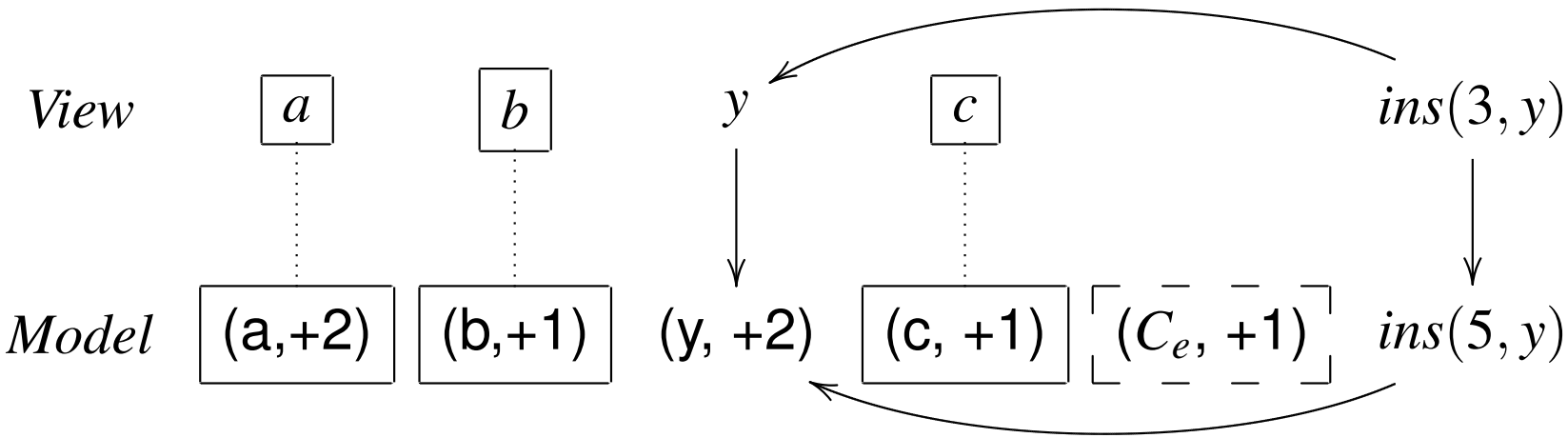
$T(ins(p_1, c_1, sid_1), del(p_2, sid_2))$ :−
**return** $ins(p_1, c_1, sid_1)$

$T(del(p_1, sid_1), del(p_2, sid_2))$ :−
**return** $del(p_1, sid_1)$

# TTF Delta Model...




(a) TTF uncompacted model



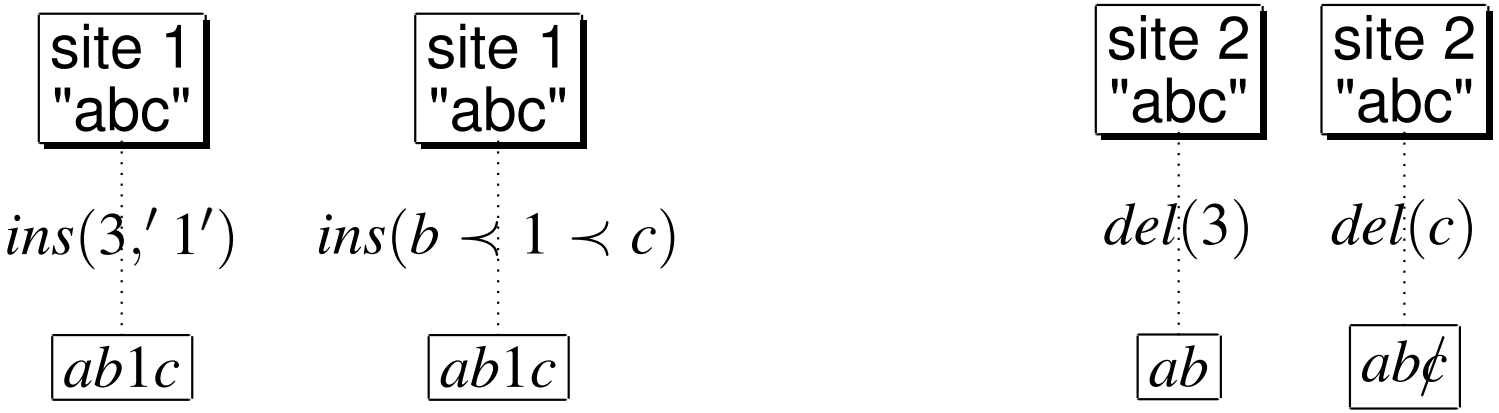
# Limitations TTF...

- Les algorithmes de transformées doivent détecter la concurrence.
- Utilisation de vecteurs d'états proportionnels aux nombres de sites.
- Le nombre de site peut devenir très grand.
- Problème de passage à l'échelle de l'approche.

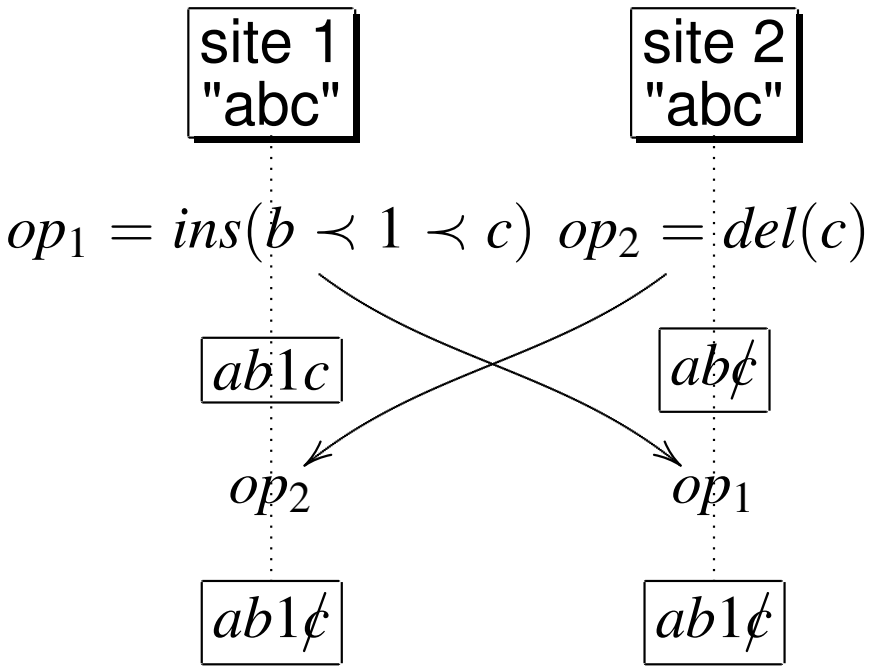
# Plan de la présentation

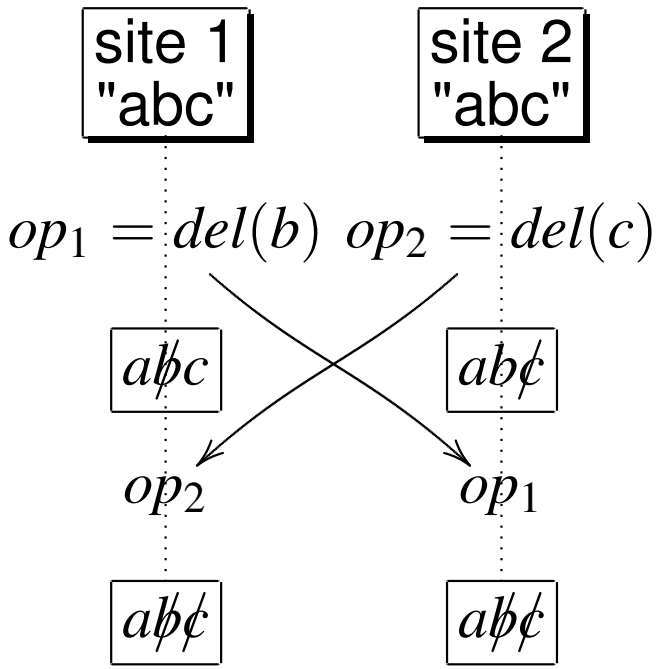
# WOOT

- Modifier le profil des opérations :
  - $ins(p \prec c \prec n)$ : insère le caractère $c$ entre $p$ et $n$
  - $del(c)$ : supprime le caractère $c$



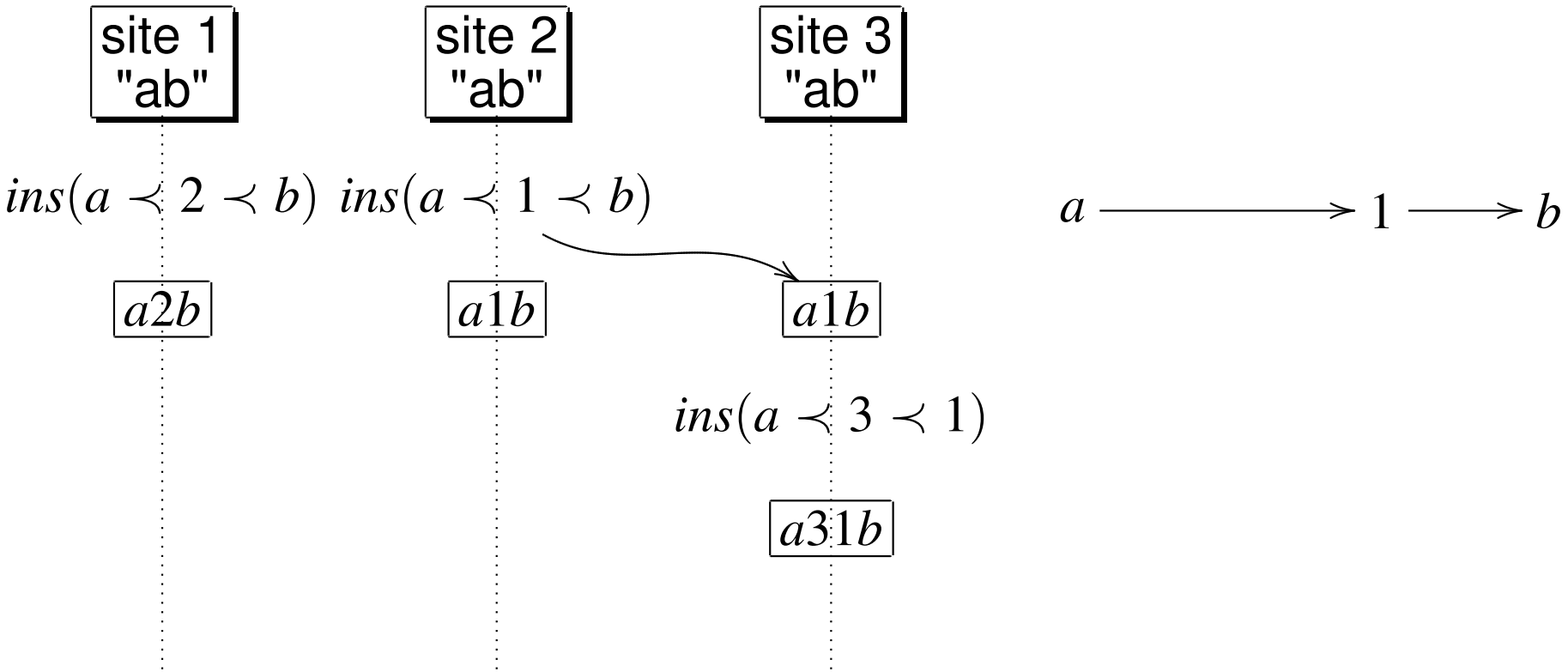
# WOOT : Commutativité



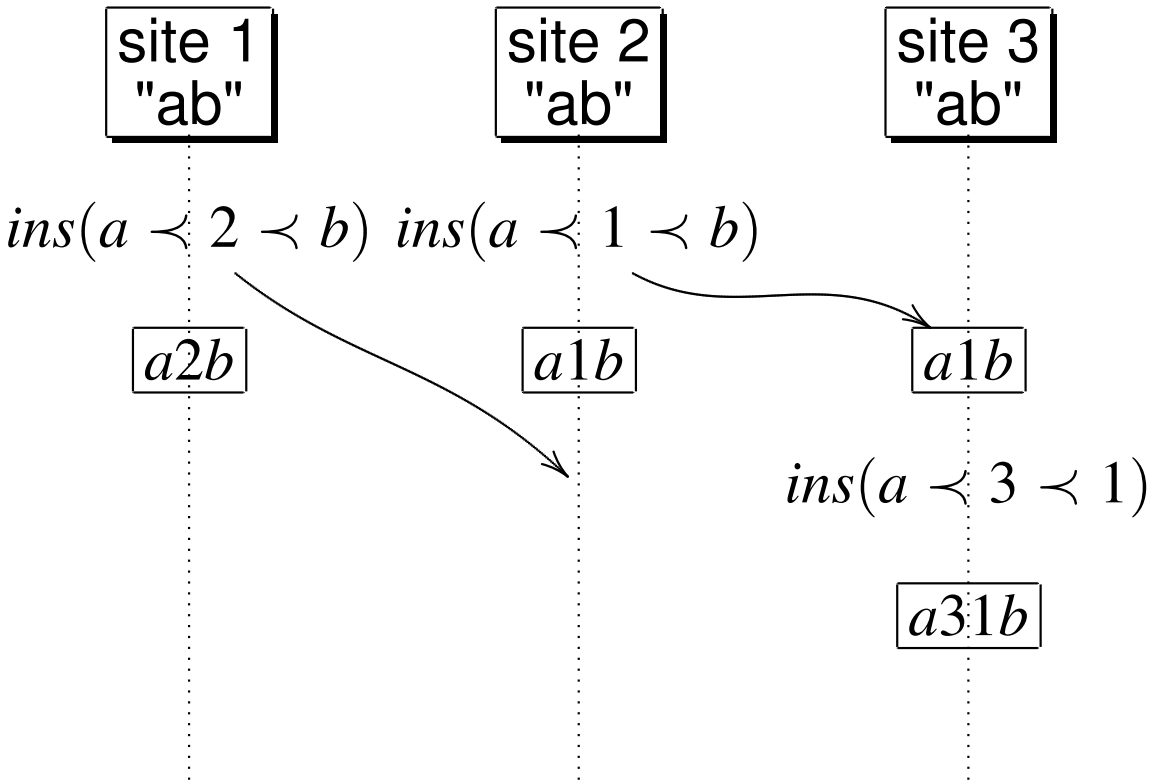



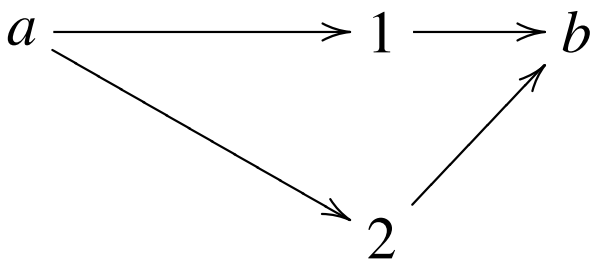

- $del()/ins()$ commutent
- $del()/del()$ commutent
- $\Longrightarrow$ $ins()/ins()$ à rendre commutatif

# WOOT:Problème Ins/Ins

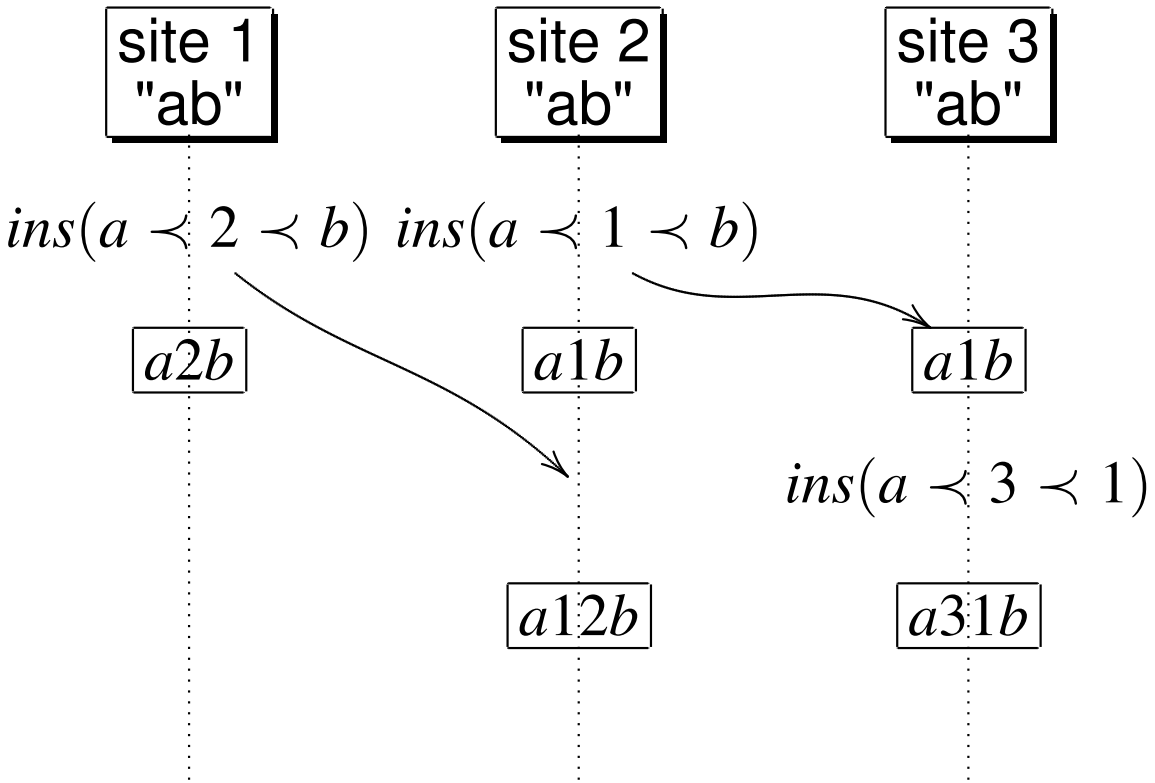


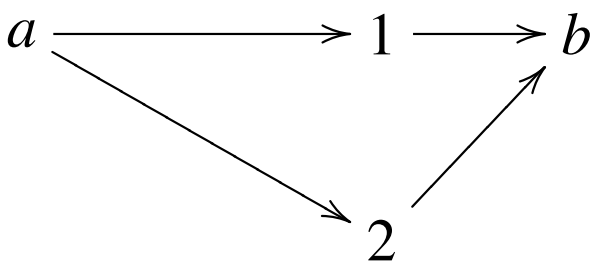
# WOOT:Problème Ins/Ins



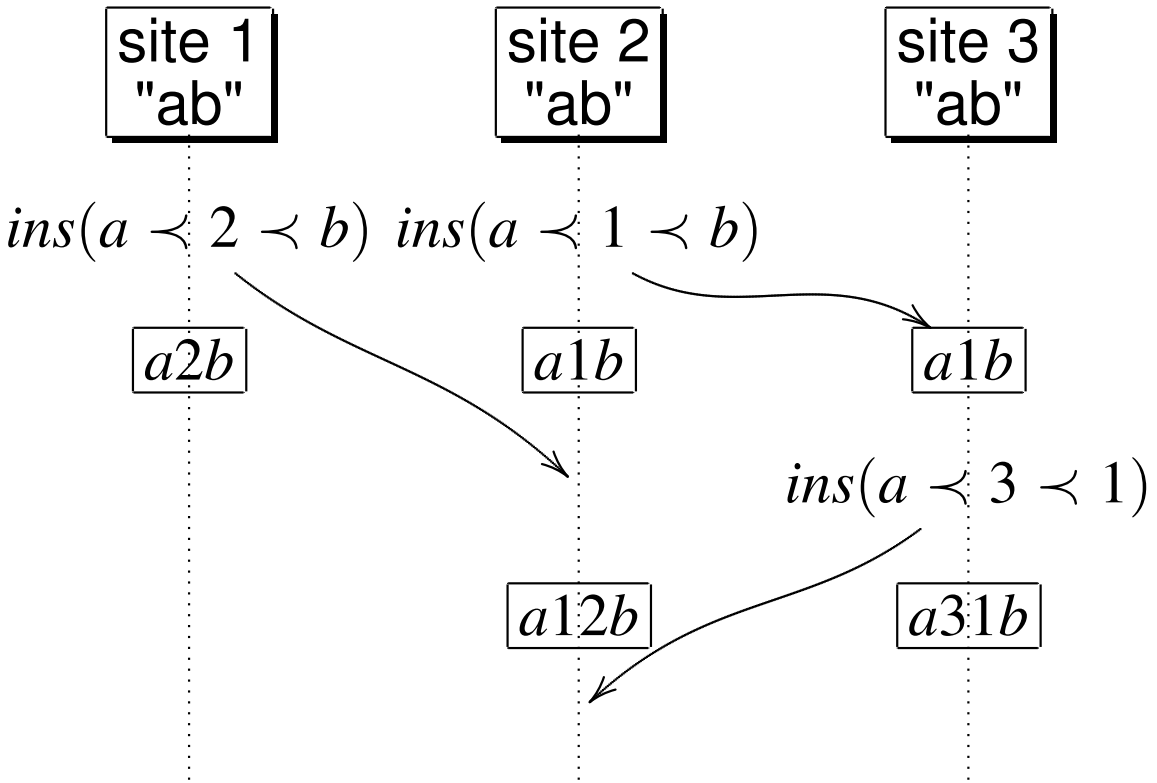



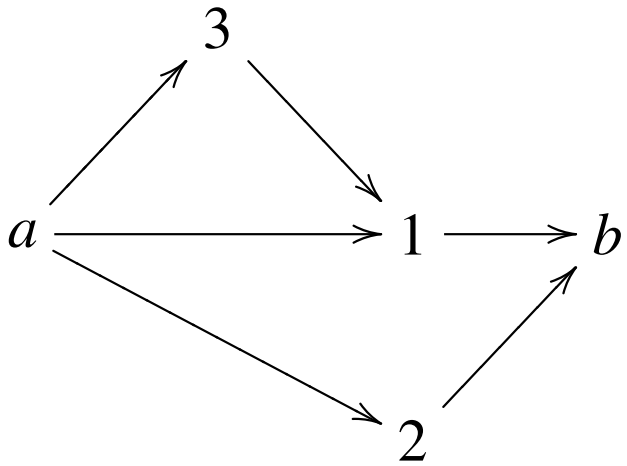
# WOOT:Problème Ins/Ins



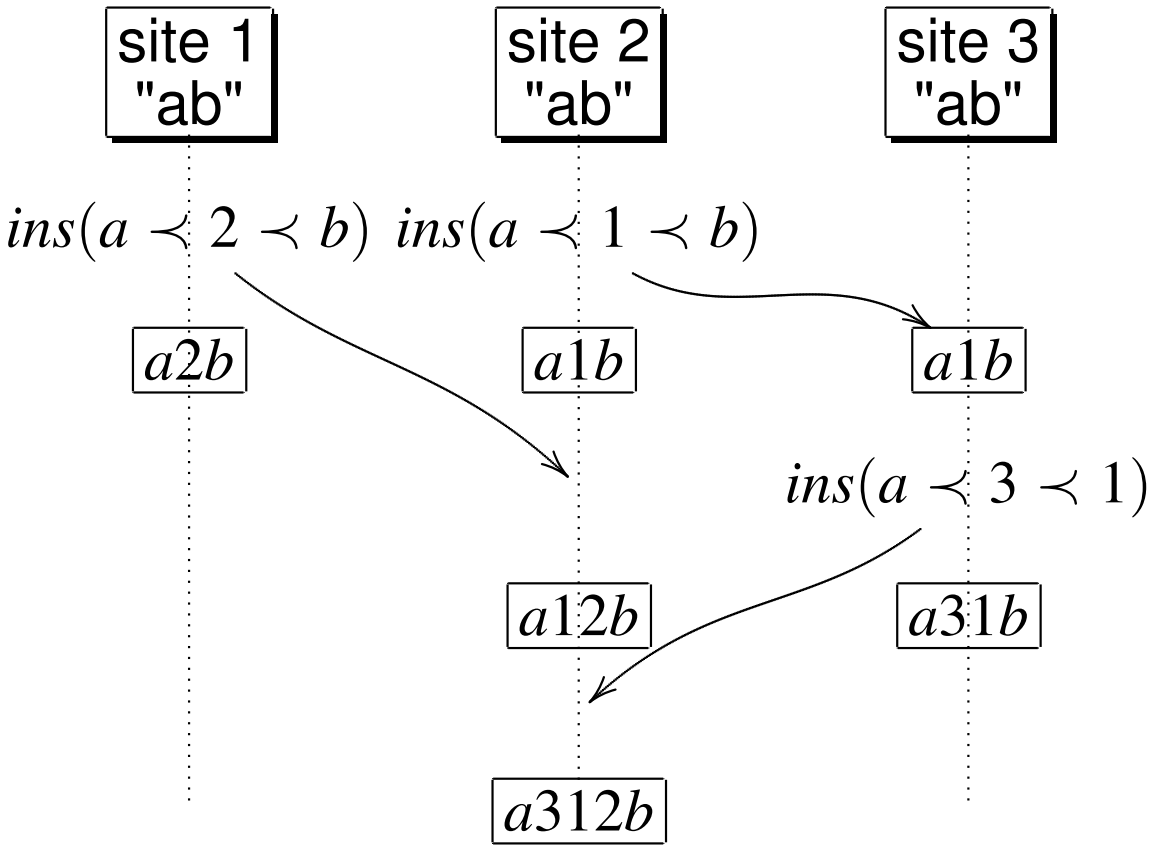



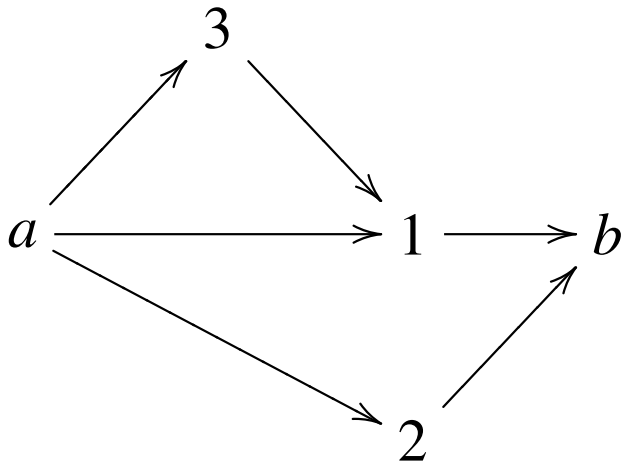
# WOOT:Problème Ins/Ins



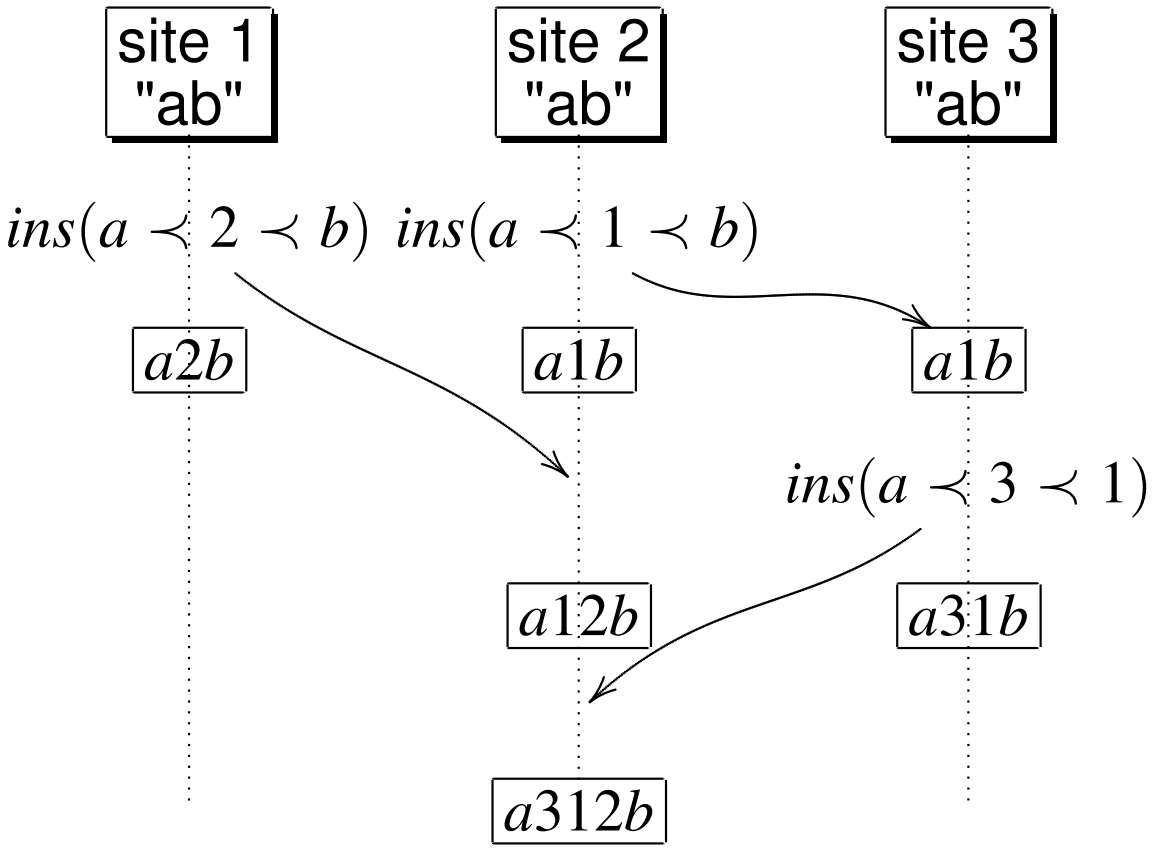



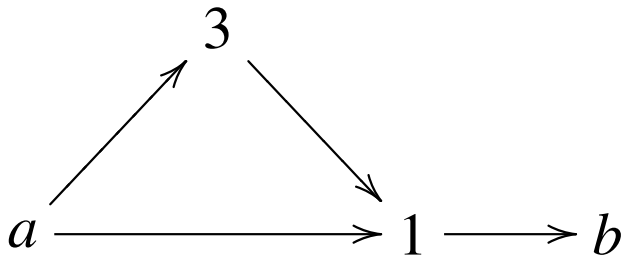
# WOOT:Problème Ins/Ins



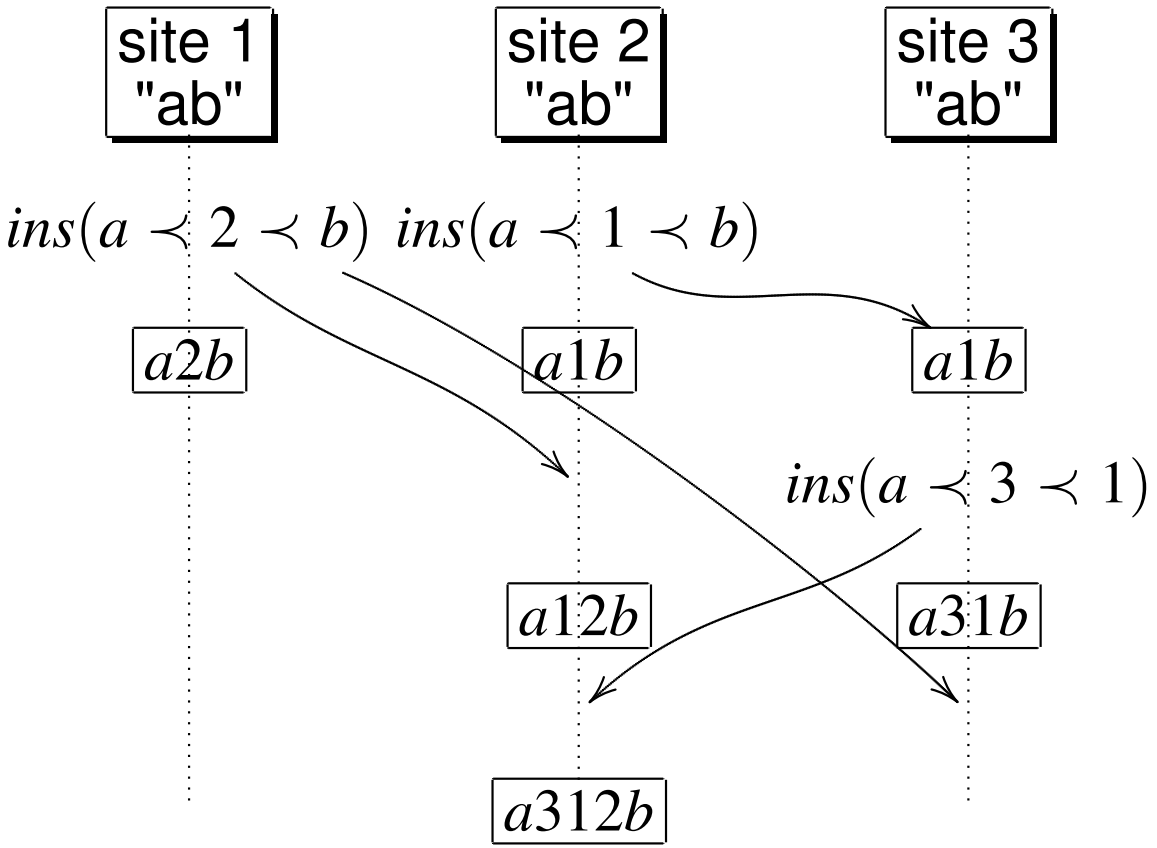



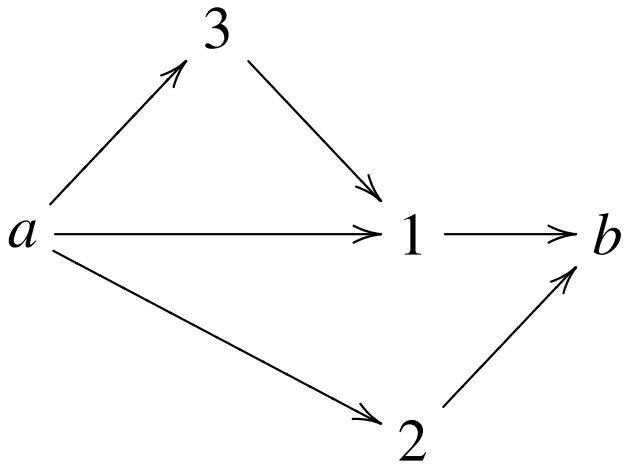
# WOOT:Problème Ins/Ins



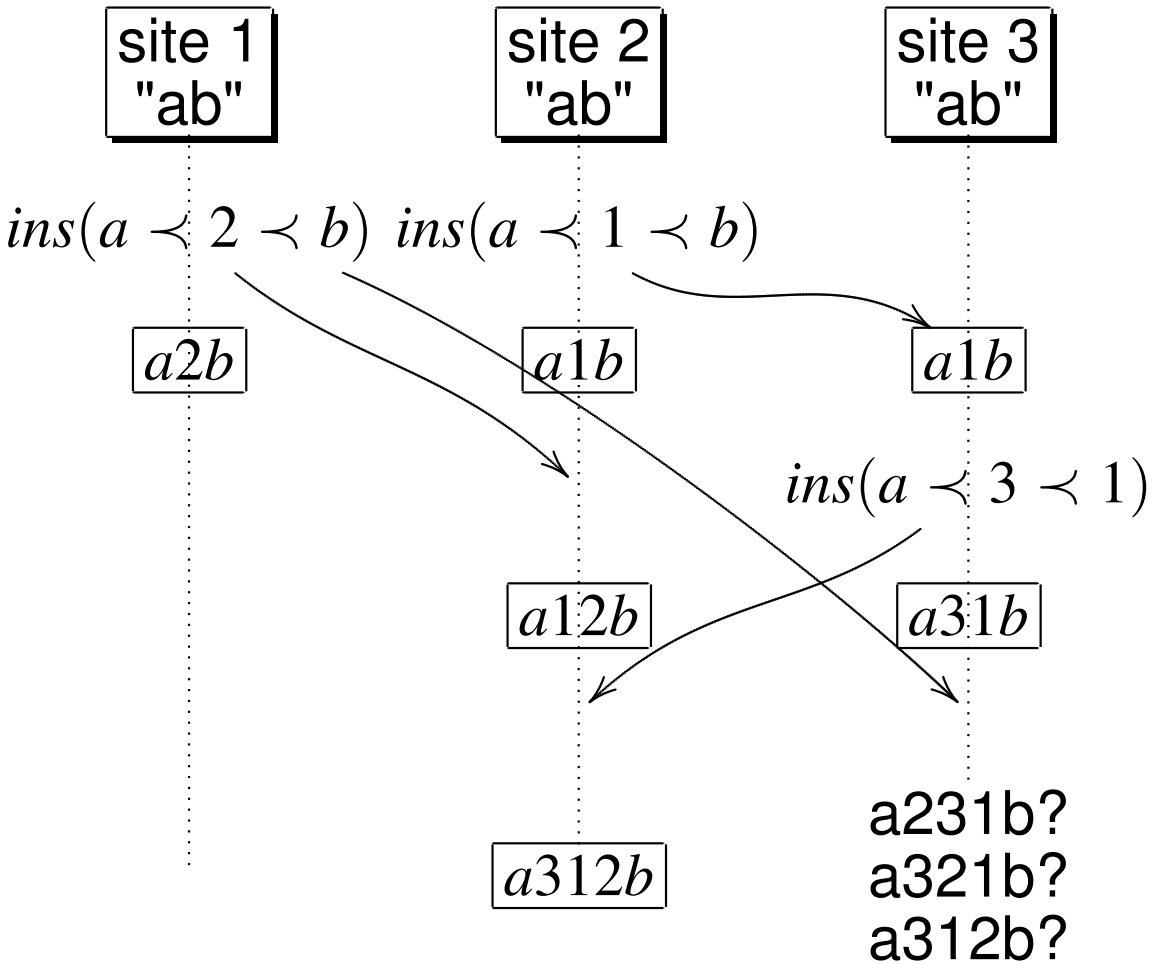



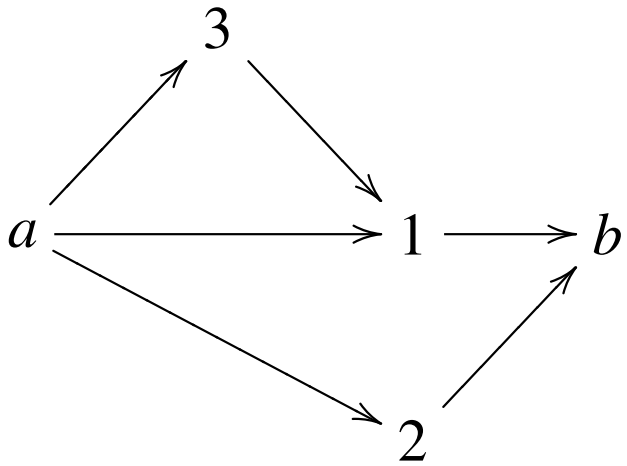
# WOOT:Problème Ins/Ins



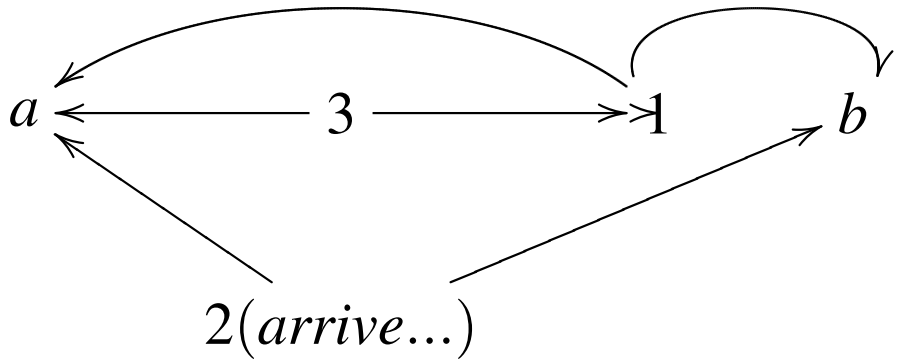



# WOOT:Problème Ins/Ins



# Idée de l'algorithme de WOOT




- Comparer les nouveaux caractères aux caractères concurrents en suivant l'ordre causal...
- Les caractères concurrents sont les caractères apparaissant entre les relations de la nouvelle opération
- Ordre causal : 1 est arrivé avant 3
- Donc on compare '2' d'abord avec '1'

# WOOT : Correction

- algorithme de réplication optimiste reposant sur un calcul :
  - d'une extension linéaire de $\prec$
  - indépendant des autres sites
- Nous avons vérifié :
  - la convergence (*model-checking*) (spécification TLA)

# Limitations WOOT

- Passe à l'échelle mais:
- Pas générique comme OT+TTF...
- Pas encore de preuve comme dans OT...

# Plan de la présentation

# Contributions

## Transformées opérationnelles (OT)

- SAMS : Le premier éditeur multi-synchrone...
- SO6 : un gestionnaire de configuration basé sur OT: transfert industriel (http://www.libresource.org/)
- VOTE : un environnement de conception et de vérification pour OT : Vérification de toutes les fonctions de transformation existantes...
- TTF : Les premières fonctions de transformation vérifiant $C_2$.

## WOOT

- un nouvel algorithme de réplication optimiste pour des structures linéaires qui passe à l'échelle...

# Perspectives à court terme

Encore la cohérence : gérer l'annulation de groupe...

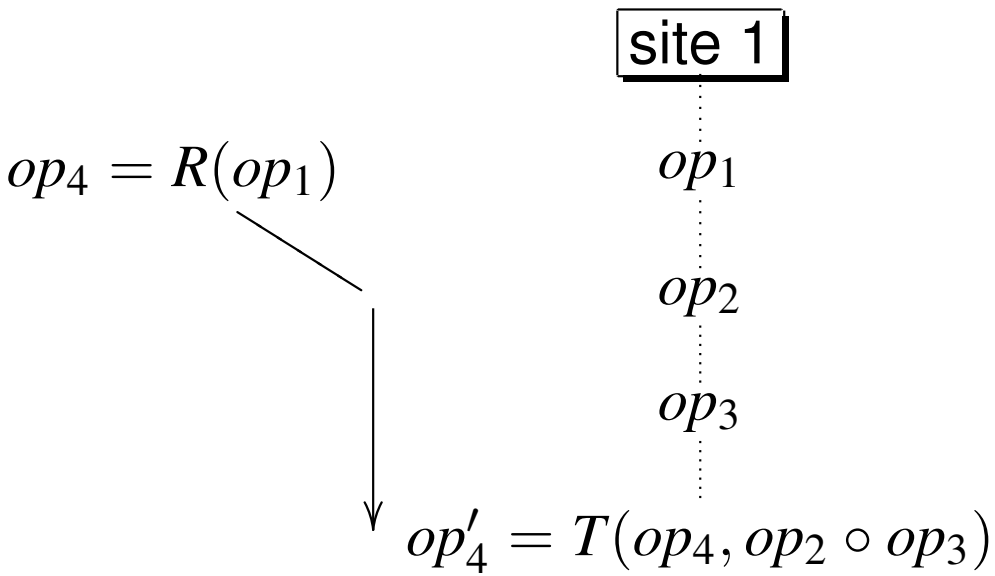
De la conscience de groupe : Gérer les conflits en P2P...

Vers des données plus complexes : Travailler avec des arbres XML, des systèmes de fichiers : combinaison TreeOpt + TTF,

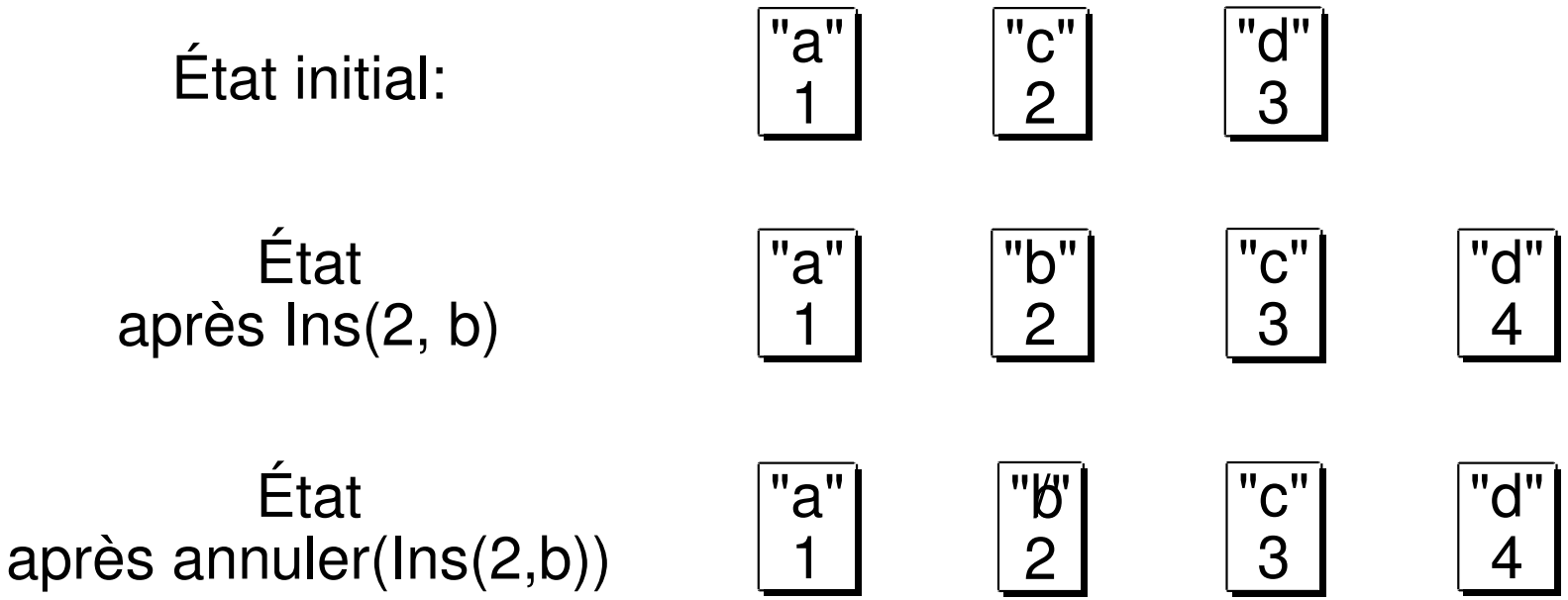
# Plan de la présentation

# Annulation de groupe

- Pouvoir annuler n'importe quelle opération n'importe quand...



# Impossibilité "d'annuler" dans les TTF




- Impossibilité de retourner à l'état initial ⇒ Prendre une approche basée sur la compensation...

# La compensation

- A chaque opération (de compensation ou non) est associée une opération de compensation,
- Ensemble des opérations de compensation fini,
- opération + son opération de compensation $\Rightarrow$ état acceptable.
- Écrire des fonctions de transformation pour toutes les opérations (y compris les opérations de compensation) vérifiant $C_1$, $C_2$ et $C_c$,

# La condition de compensation $C_c$

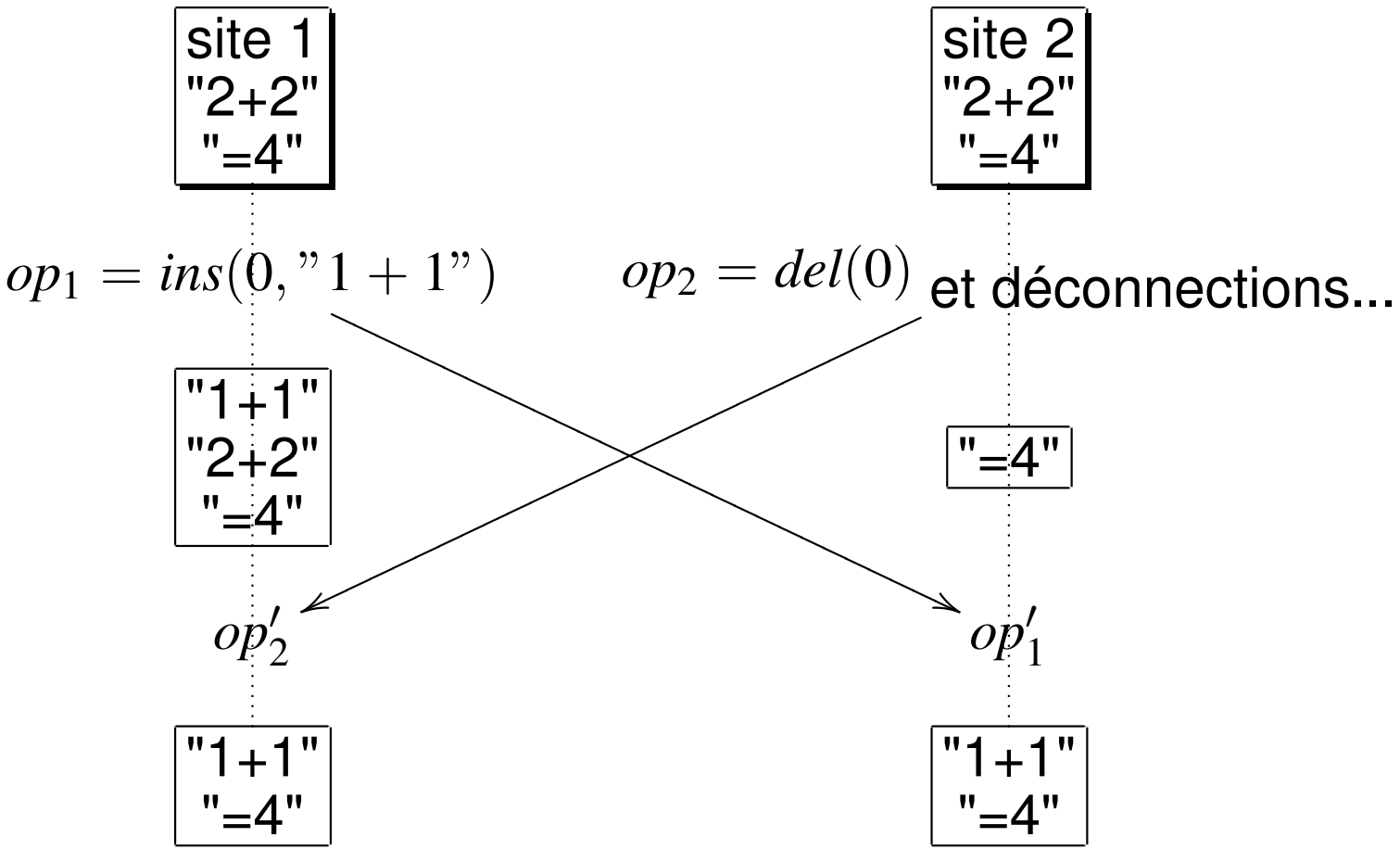
- Condition de Compensation: $C_c$
  - $T(R(op), T(seq, op)) = R(T(op, seq))$
  - Avec $op$ l'opération que l'on souhaite compenser, $seq$ une séquence d'opération et $R(op)$ la fonction qui associe une opération de compensation à l'opération $op$.
- Écritures des fonctions TTF avec opérations de compensation + vérification des $C_1$, $C_2$, $C_c$ par VOTE...

# Plan de la présentation

# Gestion des conflits en environnement P2P

- Dans 1 modèle copier-modifier-fusionner, la dernière version est toujours produite par un humain...
- En P2P, 2 utilisateurs peuvent produire des modifications concurrentes et le temps de propagation dans le réseau P2P peut faire en sorte que ce soit le moteur de synchronisation qui génère l'état visible.

# Exemple


site 1
"2+2"
"=4"
site 2
"2+2"
"=4"
$op_1 = ins(0, "1+1")$
$op_2 = del(0)$
et déconnections...
"1+1"
"2+2"
"=4"
"=4"
$op'_2$
$op'_1$
"1+1"
"=4"
"1+1"
"=4"

# Gestion des conflits en réseaux P2P

- Comment avertir les utilisateurs que cette page a été générée automatiquement ?
- Comment mettre en avant ou sont exactement situés les changements conflictuels sans avoir à relire tout le document ?

# Perspectives

**WikiWikiWeb** Wikipedia, Citizendium, Wikia, Wiki d'entreprise → *Collaboration massive...*

**Gestion de configuration** : Clearcase, CVS [1986], puis Subversion [2001], Arch, Darcs [2005], Baazar [2005], Monotone, GIT [2005], Mercurial [2005] → *Collaboration décentralisée, P2P*

**Éditeur en ligne** : Google Docs and Spreadsheets [Sep 2006], Zoho suite [2007], EditGrid [2006], Firefox3 → *Outils collaboratifs on-line*

**Éditeur temps réel** : Co-word [2005], Subethaedit, Gobby [2005], ACE [2005] → *Collaboration temps-réel*

## Fusion à terme

- Environnement de collaboration massive, P2P, on-line, multi-synchrone...