Plan de l'exposé

- Créer une topologie calculable
- Traiter les données semi-structurées
- Parallélisation des calculs
- **Intégration du temps**
  - Réécritures de textes et structure d’événements
  - Transposition modale
  - Tetris Editor (TED)
- Conclusion & Perspectives

Réécriture et repentir

- Là où l'écriture manuelle est générative, **l'écriture mécanisée est un acte transformationnel**.
- Au cours du temps, tout texte imprimé est constamment travaillé, sur les épreuves, jusqu'à la publication.
- Avec le support informatique, **les épreuves sont innombrables et recyclables** à coût marginal faible et sans traces.
- Tout texte publié est **la partie visible d'un processus de création décentralisé** et coopératif.

Les illusions perdues, épreuve corrigée par Balzac.
http://www.5-items.fr/formalisation.html

http://www.pourlascience.fr/ewb_pages/a/article-crire-et-se-repentir-35425.php

La réécriture (informatique)

La réécriture est un modèle utilisé en informatique, en algèbre, en logique et en linguistique.

Il s'agit de transformer des données en appliquant des règles de transformation :

- Données non-structurées : mots, textes en langages artificiels (« *codes sources* »), ...
- Données structurées : formulaires, abaques, textes en langages naturels (« *documents* »), XML, ...

Exemples d'utilisation de la réécriture :

- définir la structure d'un énoncé (syntaxe)
- exprimer le sens d'un énoncé (sémantique)
- expliciter une preuve (démonstration)

https://fr.wikipedia.org/wiki/Réécriture_(informatique)

Réécriture décentralisée

- Tout utilisateur réécrit le même document :
  1. Un premier ajoute « *le chat mange la souris* »
  2. Un deuxième change *« le verbe* » en « *aime* »
  3. Un troisième change *« l'objet* » en « *mouche* »
  4. Le deuxième accepte le « *changement d'objet* »
  5. Le premier accepte « *les changements* »
- Chaque utilisateur perçoit des énoncés « *autonomes* »
  - Quand tous les énoncés sont « *identiques* »
    → « *consensus syntaxique* » (comme après 5.)

Changer « *le verbe* » en « *aime* »

- Nous pouvons transformer cette phrase en une « règle de réécriture » :

« le verbe » → « aime »

- Nous pouvons appliquer cette règle sur un contexte :

« Le chat mange la souris »→ « Le chat aime la souris »

Termes de preuves

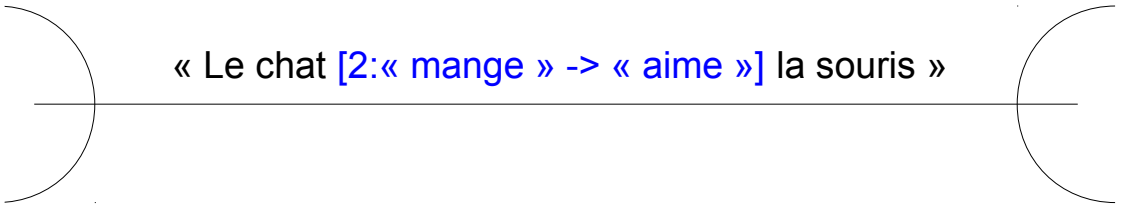
- Nous pouvons écrire l'application dans le contexte pour obtenir un terme de preuve :

« Le chat [2:« mange » → « aime »] la souris »

  - Ce terme de preuve permet de calculer la source et la cible de l'application :

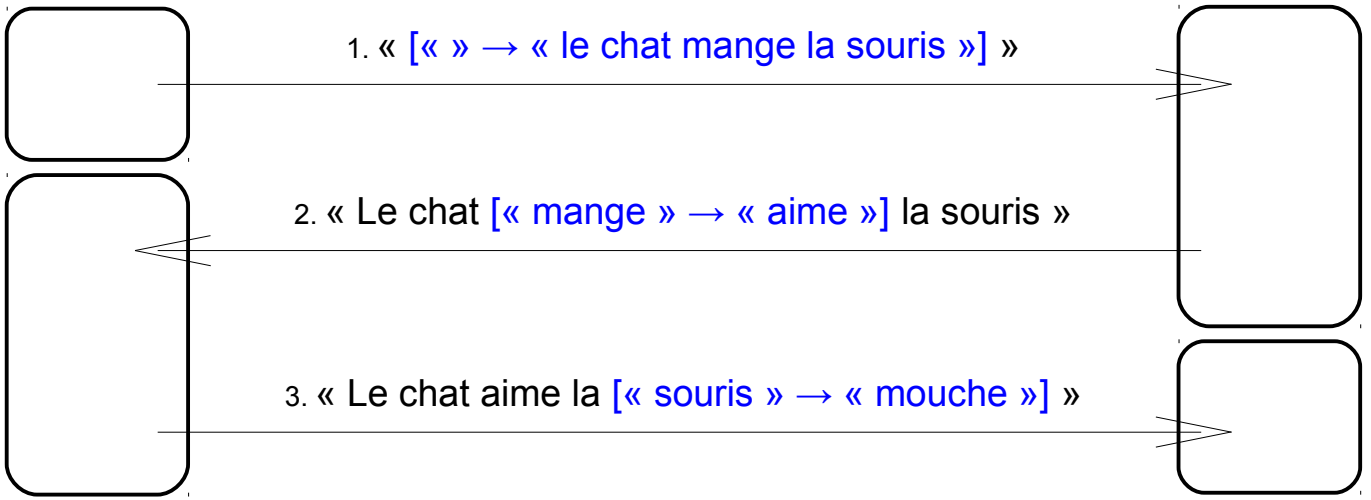
« Le chat mange la souris » → « Le chat aime la souris »

  - Ce terme de preuve permet d'écrire le graphe des états accessible en mettant l'information sur les « *arcs ouverts* »



Graphe des états accessibles

1. ajoute « *le chat mange la souris* »
2. change « *le verbe* » en « *aime* »
3. change « *l'objet* » en « *mouche* »



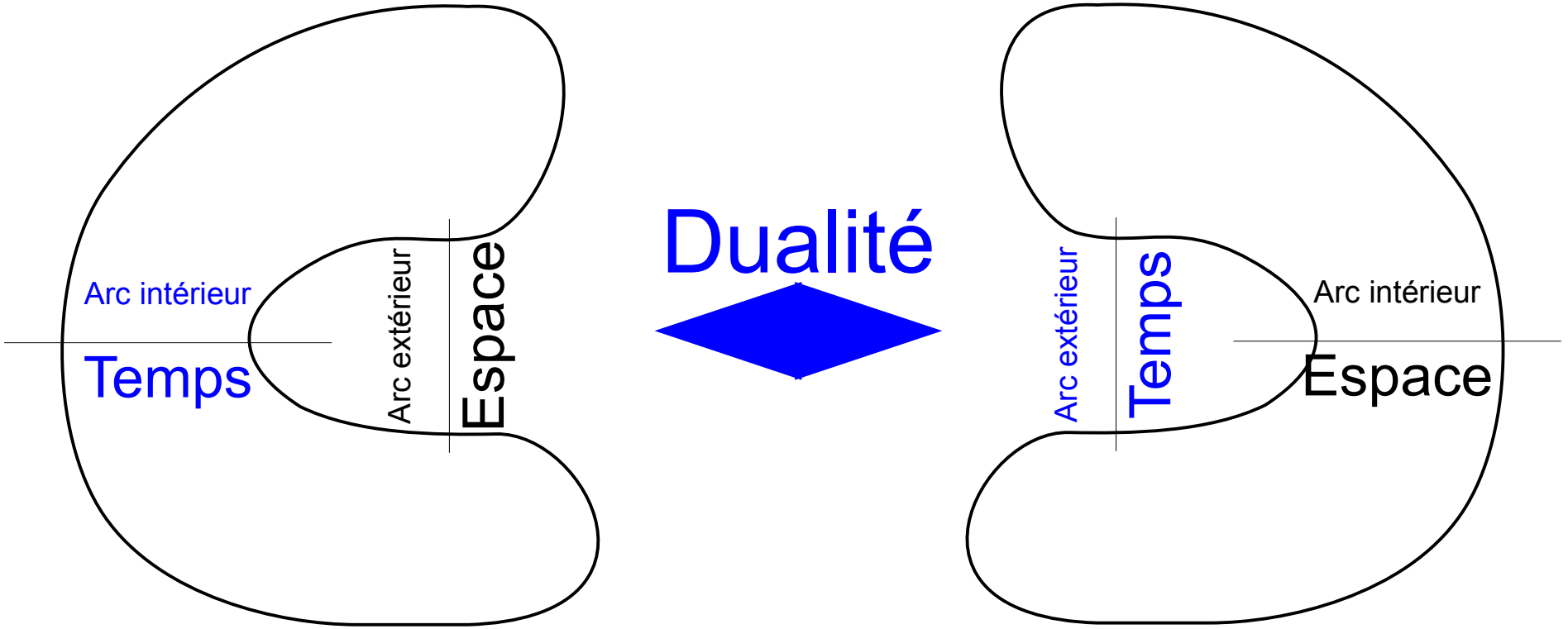
« Graphe du temps »

Intégrer le temps « *modal* »

- En formulation modale, la structure de donnée est un graphe :

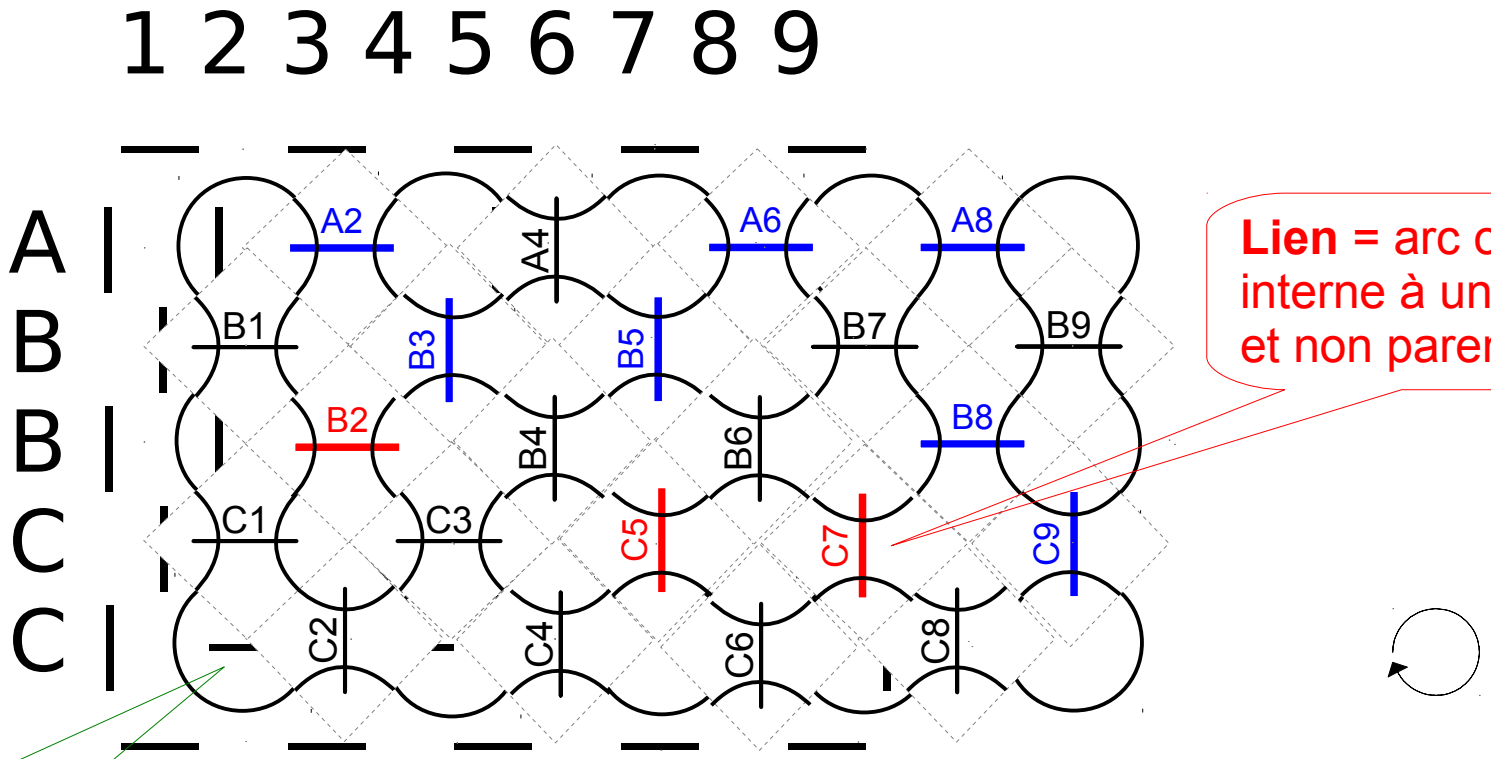
comment représenter le graphe du temps
d'un graphe de structure ?

- Nous travailler sur des graphes qui vérifient :



Ne conserver que des termes

Les arcs ouverts qui ne sont pas des parenthèses sont le seul obstacle à la dualité des dimensions



Lien = arc ouvert interne à un terme et non parenthésé

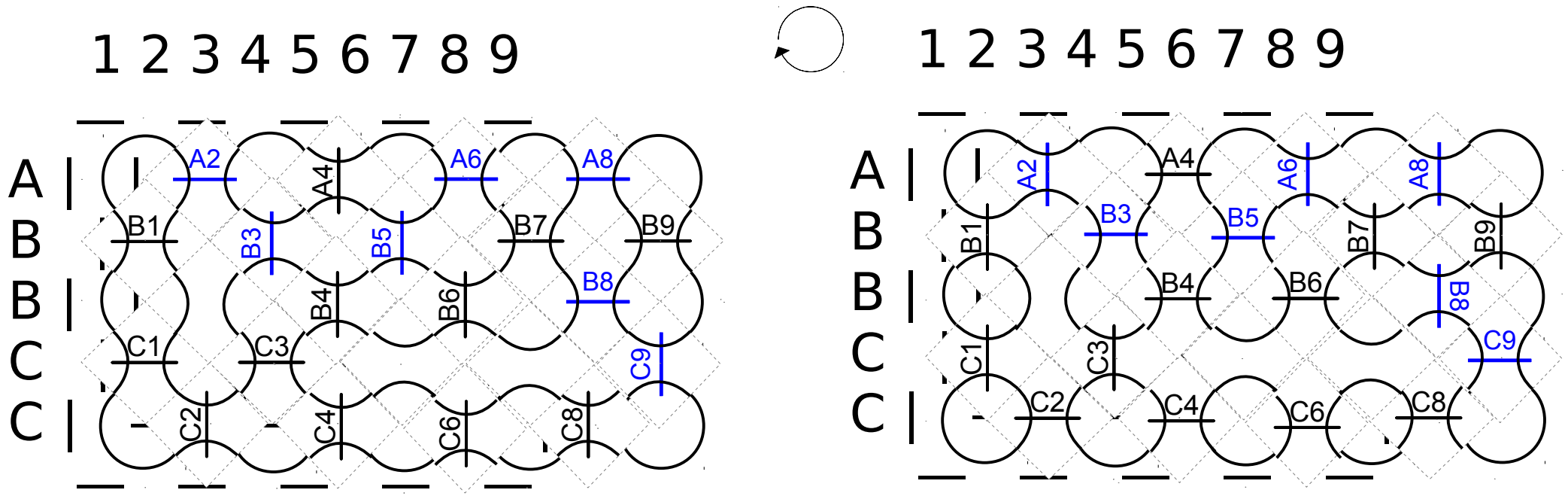
Terme = arc fermé « *bien parenthésé* »

Espace-Temps

Dualité

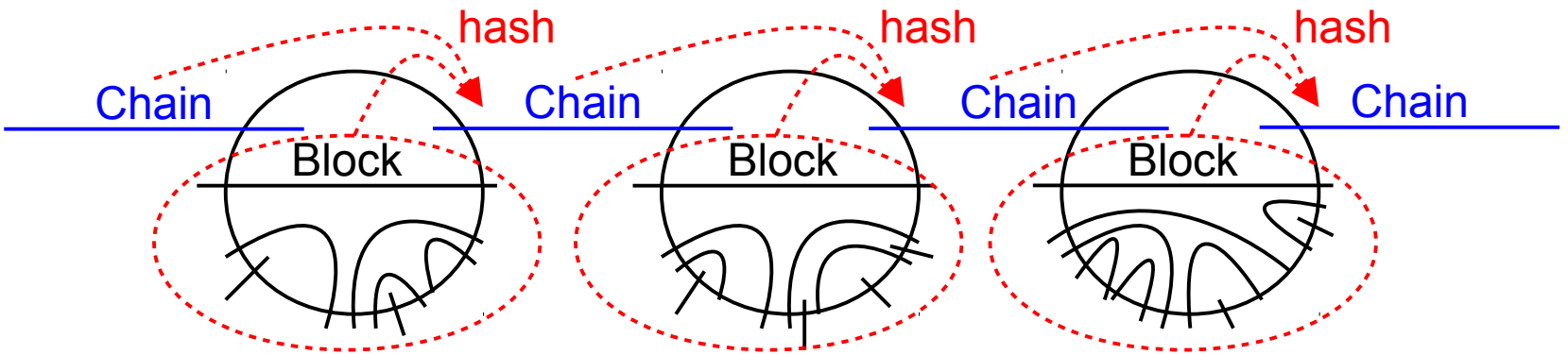
Arc intérieur → Espace
Arc extérieur → Temps

Arc extérieur → Espace
Arc intérieur → Temps

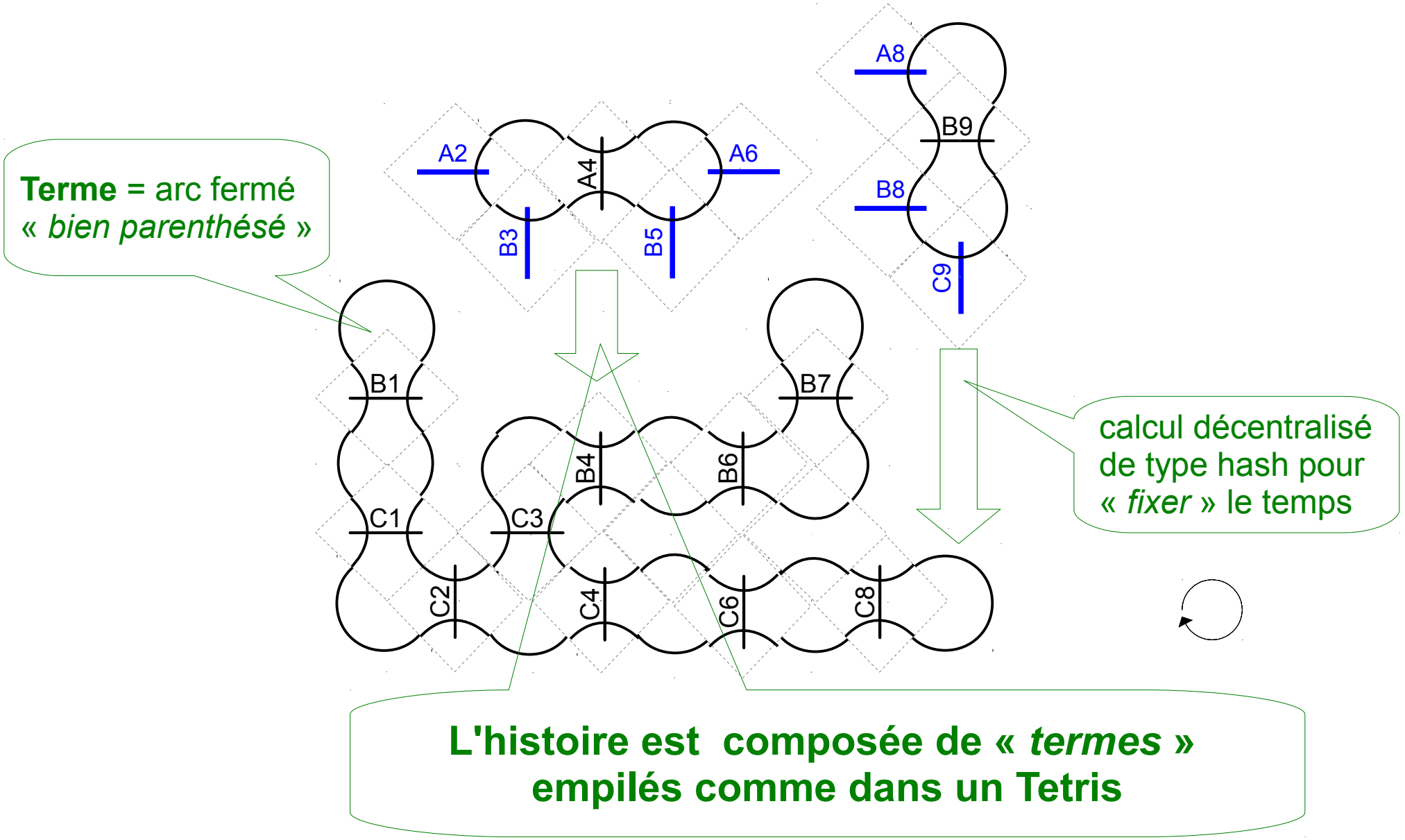


Espace Temps Blockchain

- La blockchain est un diagramme spatio-temporel
  - block = terme
  - chain = « *arc ouvert entre terme* »
- Où chaque arête temporelle contient le résultat d'un calcul cryptographique manifestant l'irréversibilité du temps.

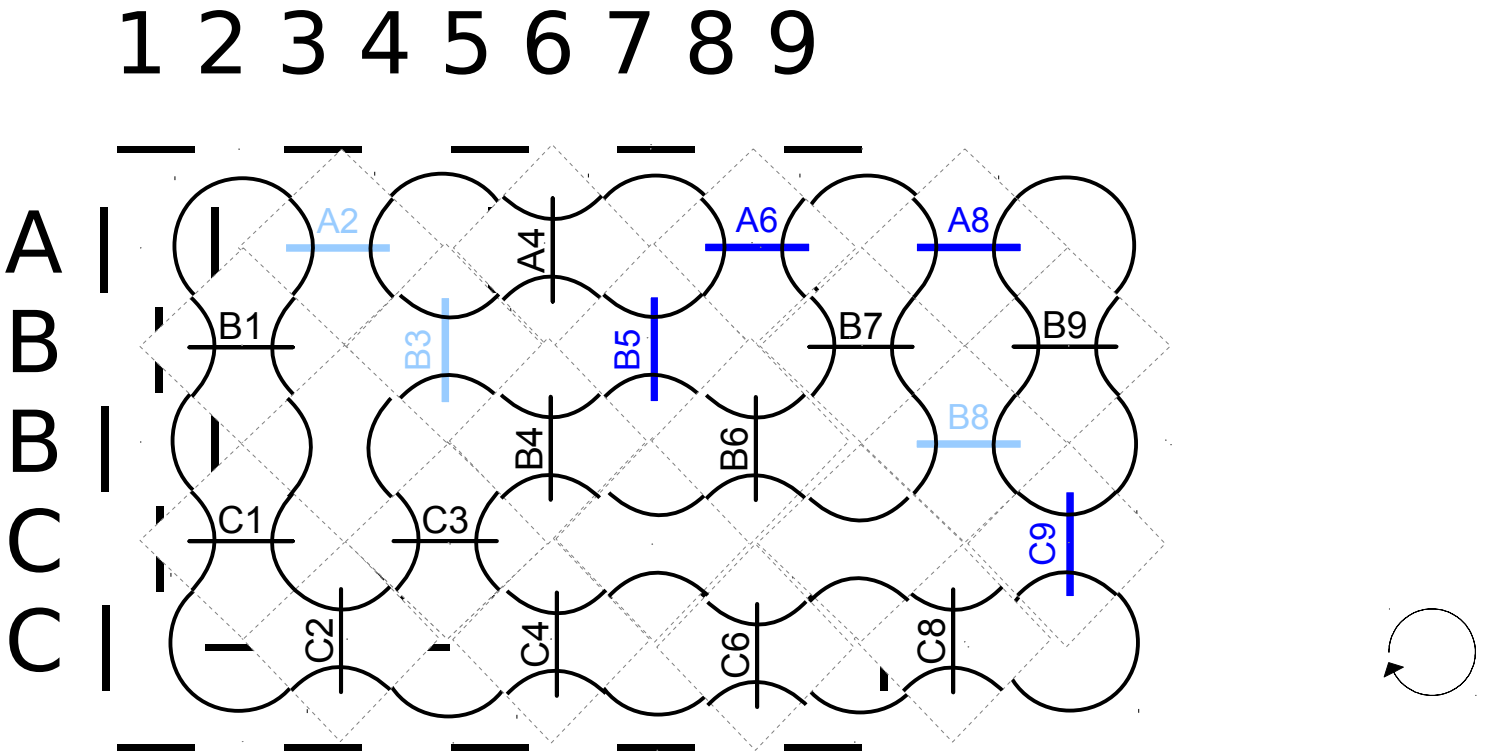


Espace-Temps Tetris



Perspective

Une perspective est un ensemble d'arcs temporels

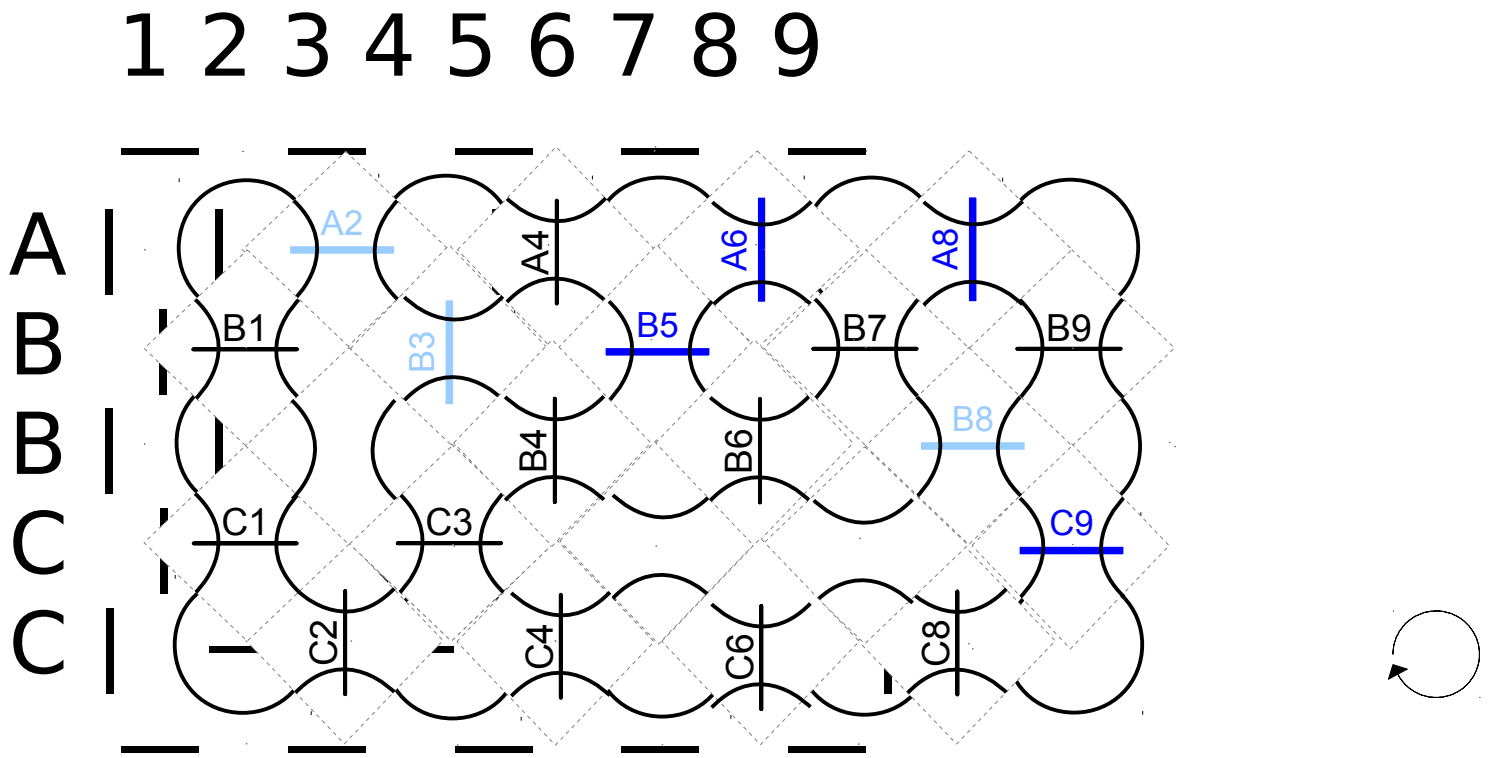


{A6, A8, B5, C9}

1415926535 8979323846 2643383279 5028841971 6939937510
5820974944 5923078164 0628620899 8628034825 3421170679

Transposition de la perspective

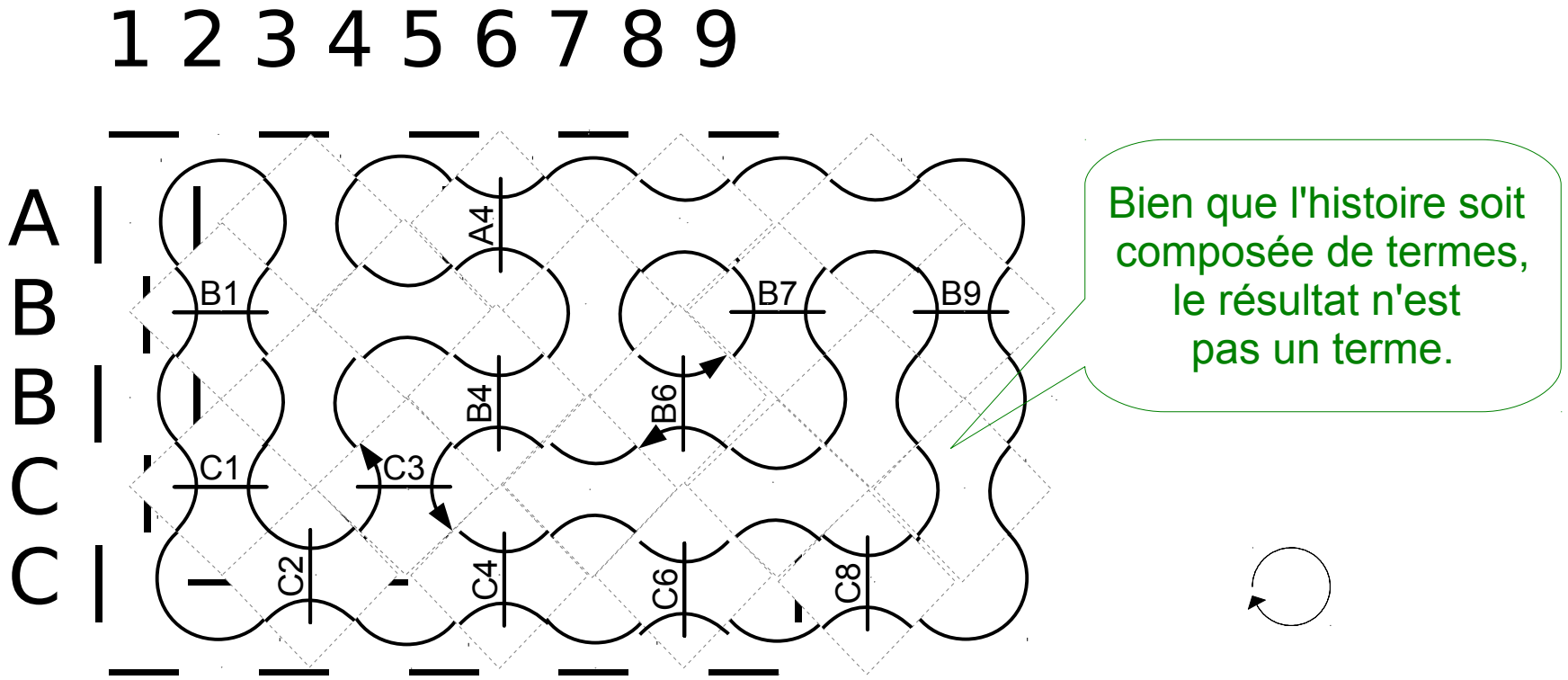
Les arcs temporels de la perspective sont transposés.



{A6, A8, B5, C9}

Calcul à l'œil du résultat

« *Oublier* » tous les arcs temporels → graphe résultat



{A6, A8, B5, C9} → {>B1, C1, C2, C3, B4, A4, A4, B9, C8, C6, C4, C2, C1, B1>, >B4, C3, C4, C6, C8, B9, B7, B6 >, >B6, B7 >}

Application à l'édition « *modale* »

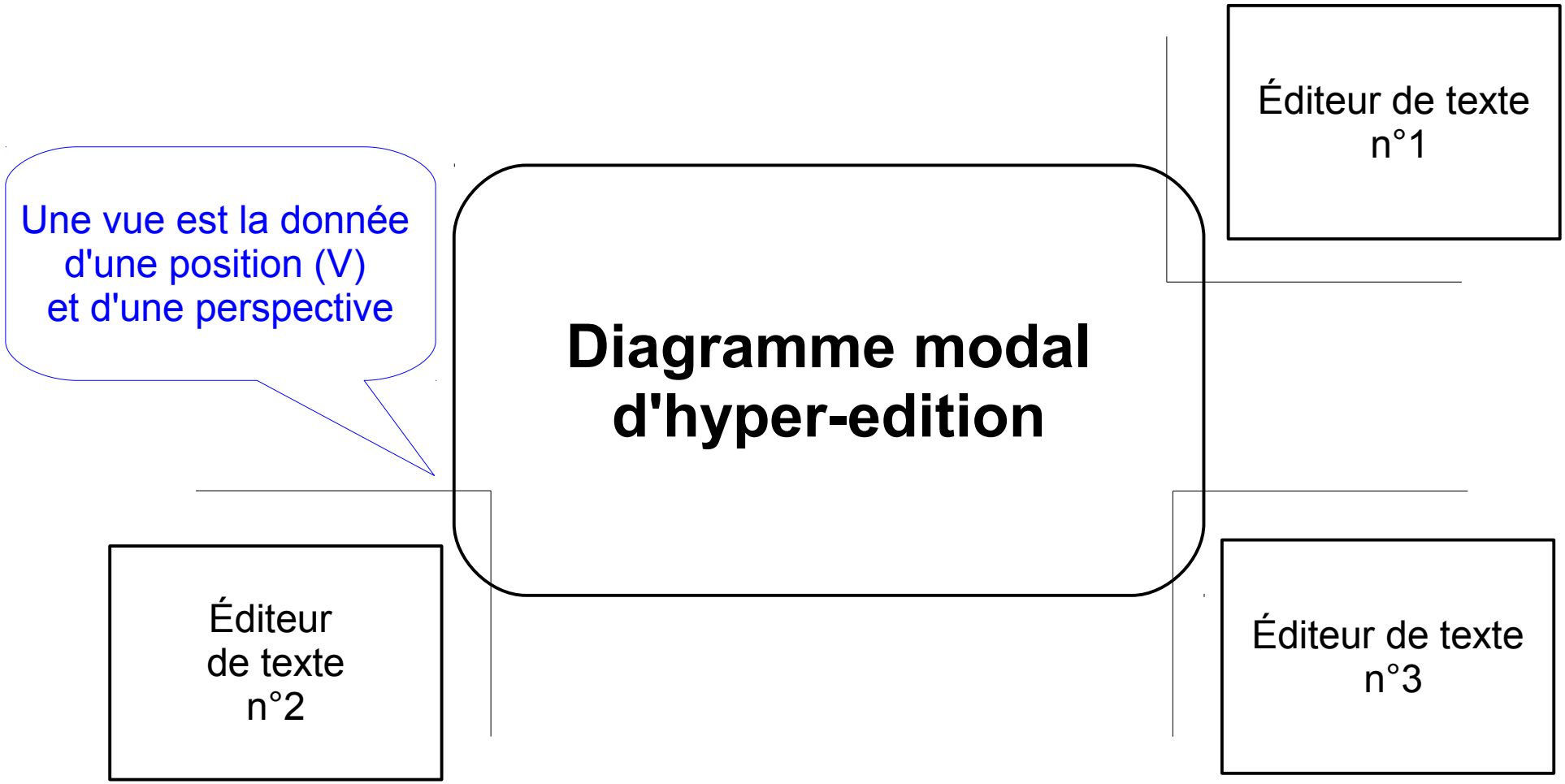
Chaque utilisateur perçoit des énoncés « *autonomes* »

1. Un premier ajoute « *le chat mange la souris* »
2. Un deuxième change *« le verbe* » en « *aime* »
3. Un troisième change *« l'objet* » en « *mouche* »
4. Le deuxième accepte le « *changement d'objet* »
5. Le premier accepte « *les changements* »

Nous allons dérouler ce cas de test avec notre « *Tetris Editor* » (TED) qui utilise l'empilage de termes comme représentation du processus d'élaboration.

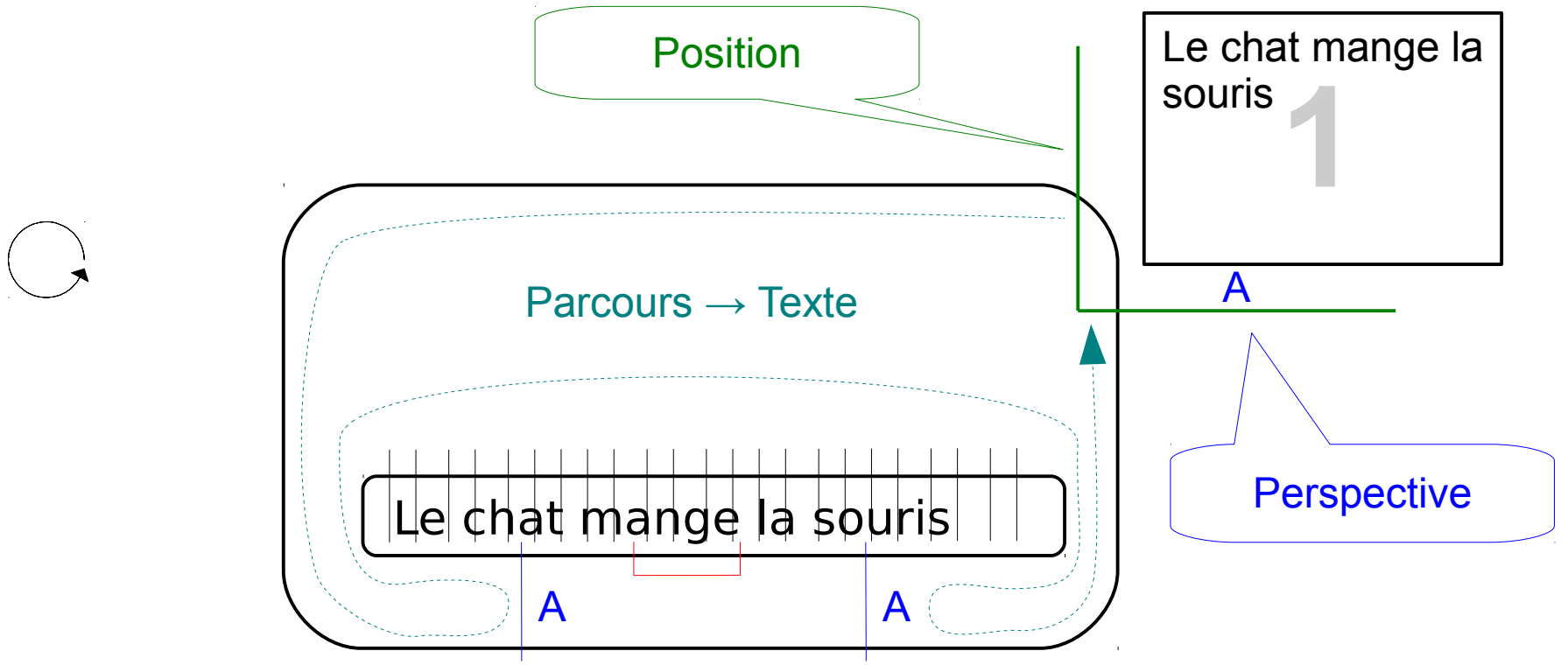
Tetris Editor

Chaque utilisateur perçoit des énoncés « *autonomes* »



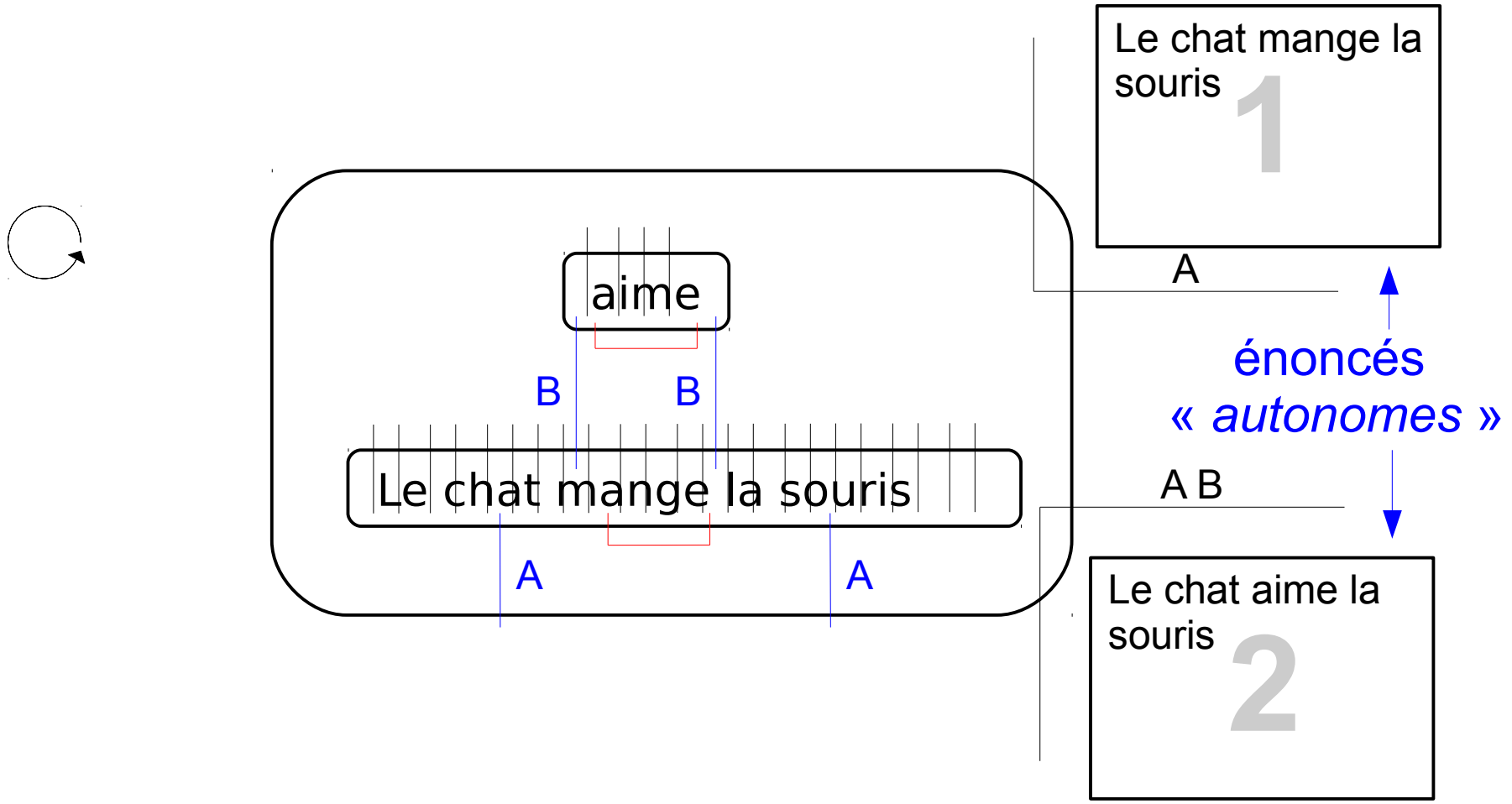
Tetris Editor

1. Un premier ajoute « *le chat mange la souris* »



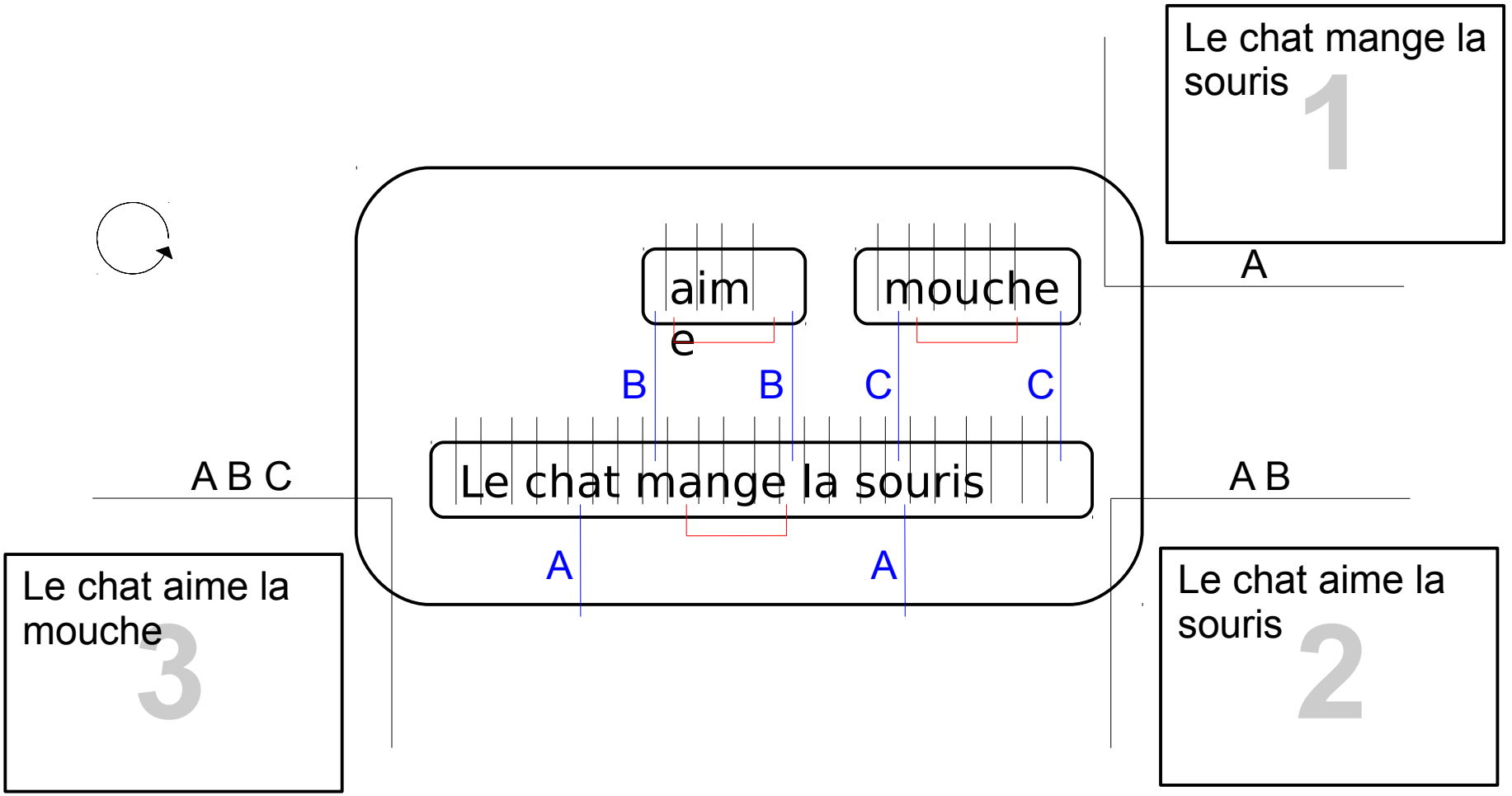
Tetris Editor

2. Un deuxième change « *le verbe* » en « *aime* »



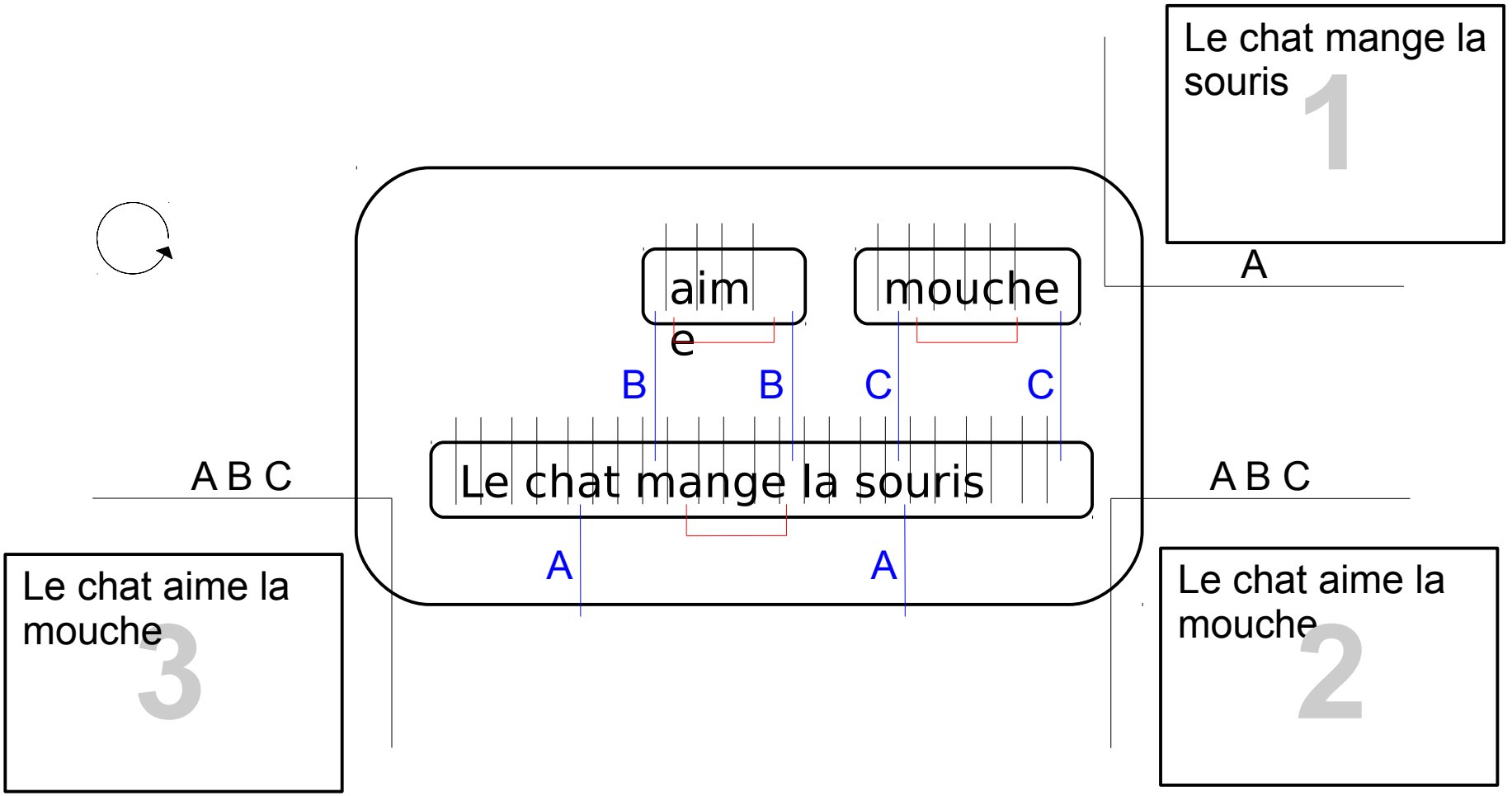
Tetris Editor

3. Un troisième change « *l'objet* » en « *mouche* »



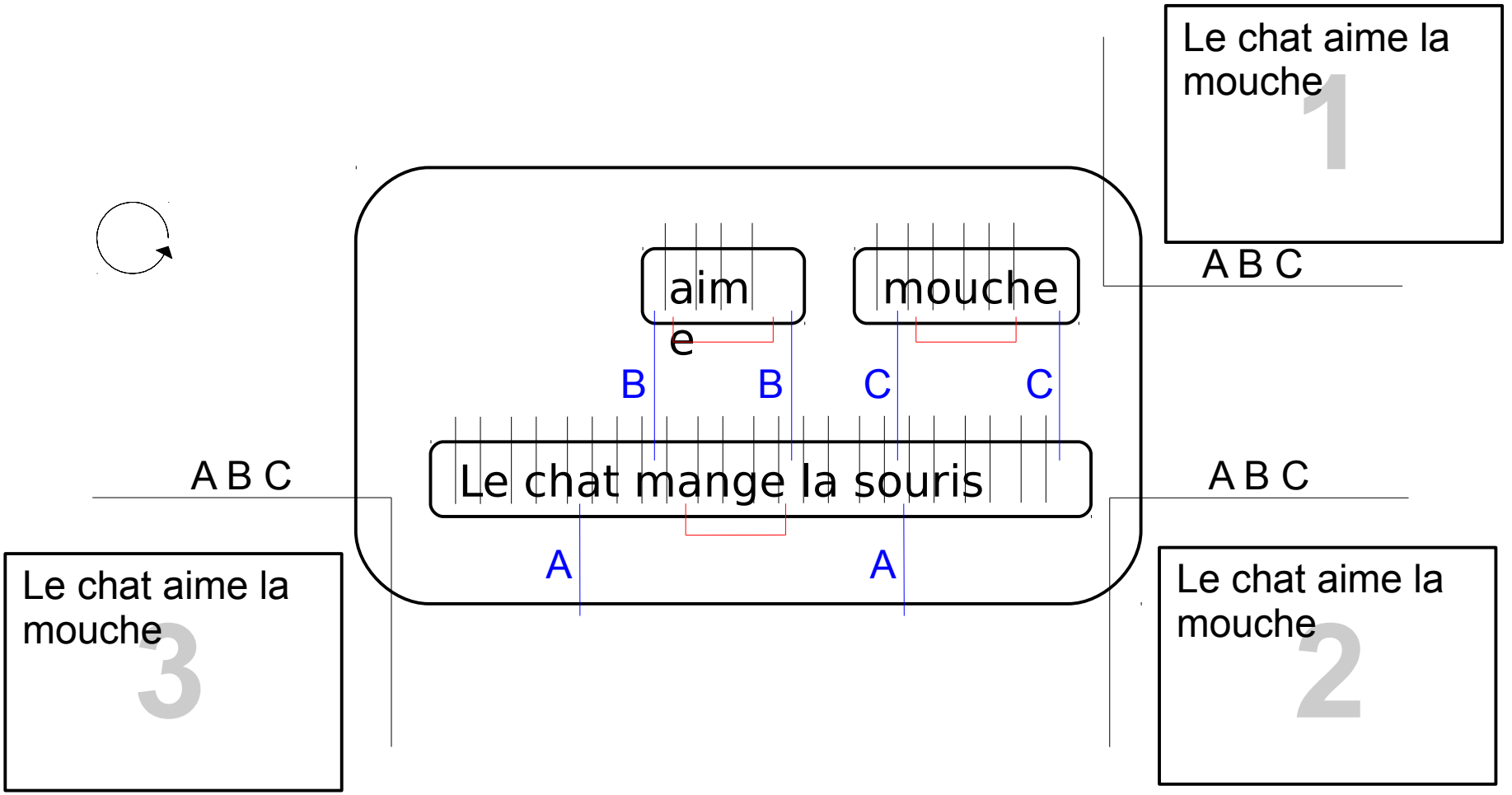
Tetris Editor

4. Le deuxième accepte le « *changement d'objet* »



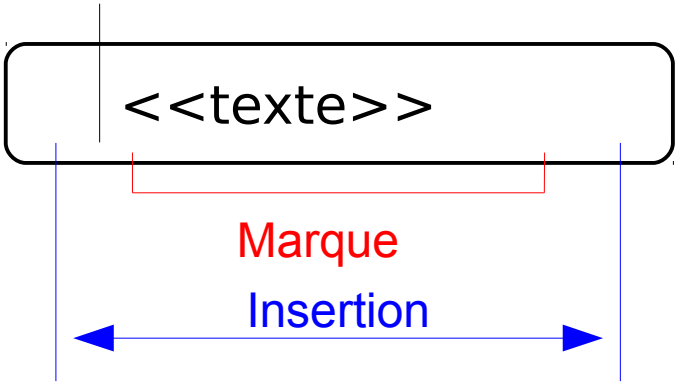
Tetris Editor

5. Le premier accepte « *les changements* »



Terme Tetris

- Un terme du « *Tetris Editor* » se compose de trois éléments :

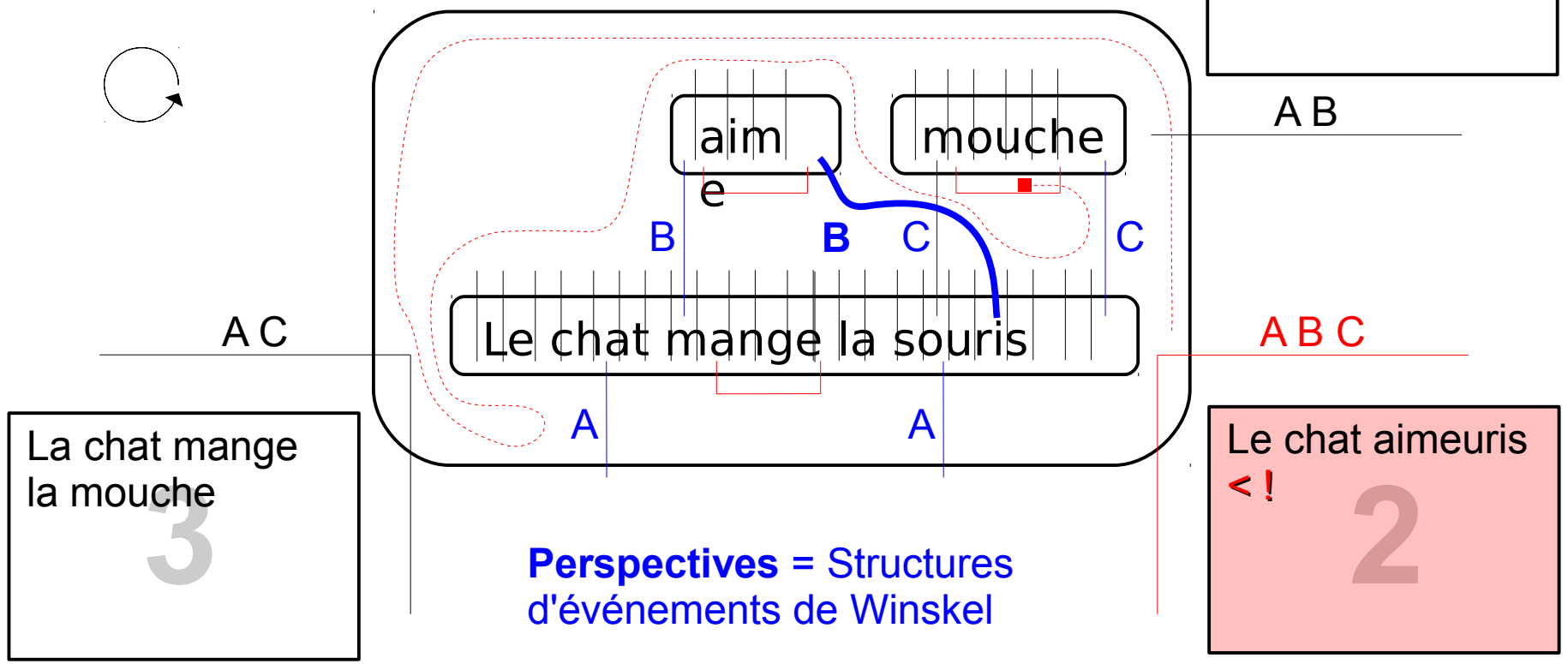


- La marque sert :
  - A détecter les inconsistances d'une vue (→ *chevauchement des pièces*)
    → Aucun parcours ne doit contenir de marques.
  - A servir de point de référence pour les pièces posées sur lui
    → Les pièces posées indique pour chaque extrémité une marque et un décalage

Perspective en conflit

Lorsque deux pièces « *se chevauchent* », certaines perspectives sont en conflit

→ le parcours contient une marque



L'histoire est un programme

- Les pièces posées indique pour chaque extrémité une marque et un décalage :
  - Un premier ajoute « *le chat mange la souris* »

    A = marque("*Le chat mange la souris*")
  - Un deuxième change *« le verbe* » en « *aime* »

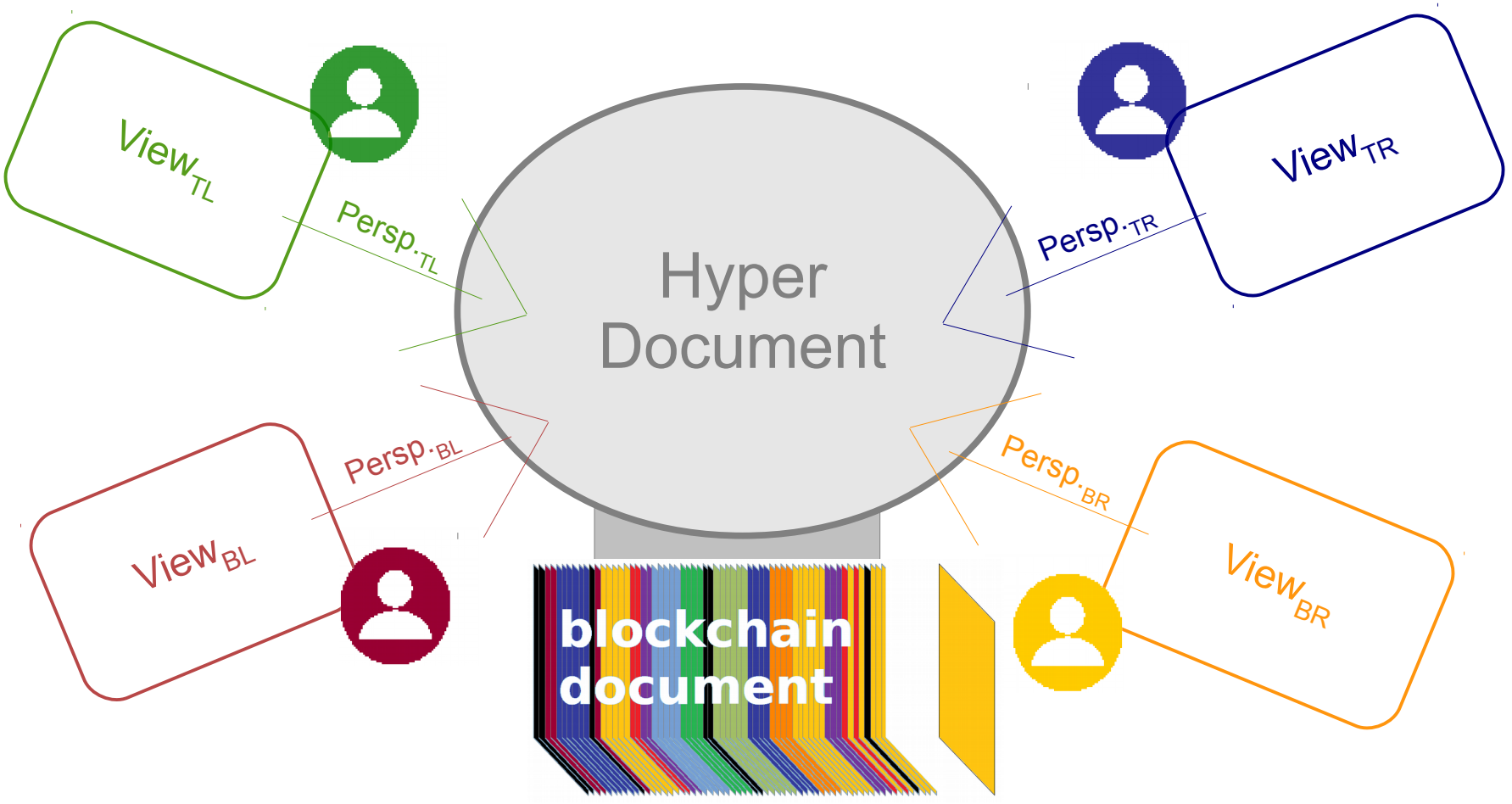
    B = marque(A[7], "*aime*",A[13])
  - Un troisième change *« l'objet* » en « *mouche* »

    C = marque(A[17], "*mouche*",A[22])
- La suite des marques forme un programme partiellement partagé entre tous les acteurs :
  - Nous pouvons équiper ce programme d'un mécanisme de type « *blockchain* » pour assurer l'inviolabilité et la provenance de chaque ligne
  - Chaque ligne de ce programme, chaque interaction de l'utilisateur avec le support d'hyper-édition, devient un acte juridique

Plan de l'exposé

- Créer une topologie calculable
- Traiter les données semi-structurées
- Parallélisation des calculs
- Intégration du temps
- **Conclusion & Perspectives**
  - Hyper-edition et diagramme espace-temps
  - Systèmes Décentralisés Interactifs (SDI)

Hyper-édition



L'hyper-document est l'espace-temps de navigation dans toute l'information du système.