Spécification des objets partagés dans les systèmes répartis sans-attente

Matthieu PERRIN

UNIVERSITÉ DE NANTES

Présentation en vue d'obtenir le grade de
Docteur de l'Université de Nantes
sous le sceau de l'Université Bretagne Loire

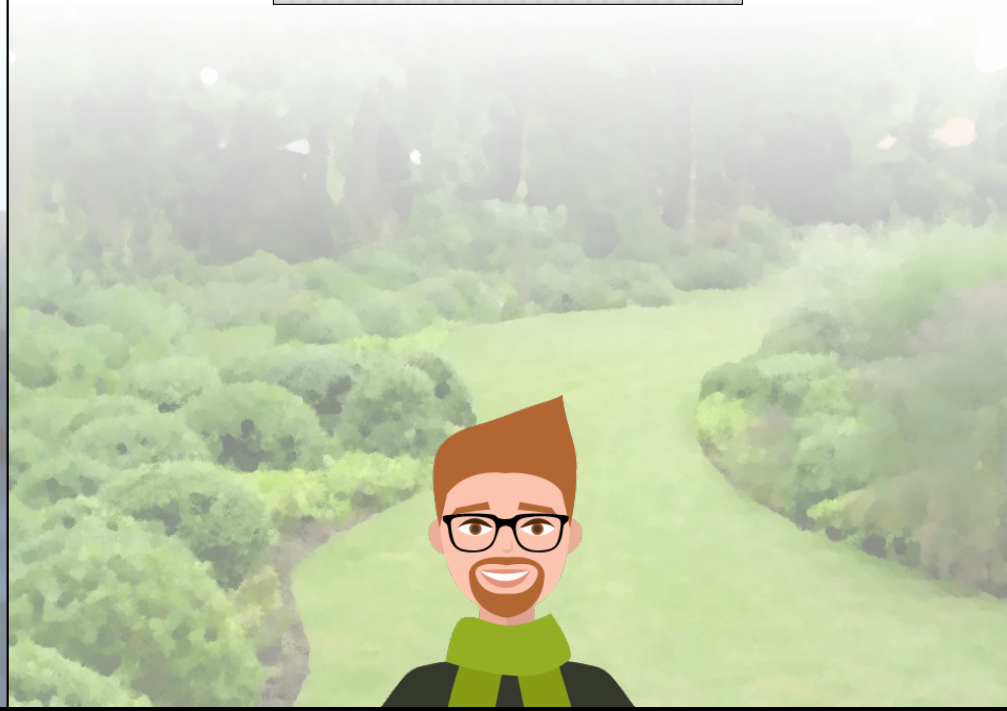
UNIVERSITE BRETAGNE LOIRE

7 juin 2016
JURY

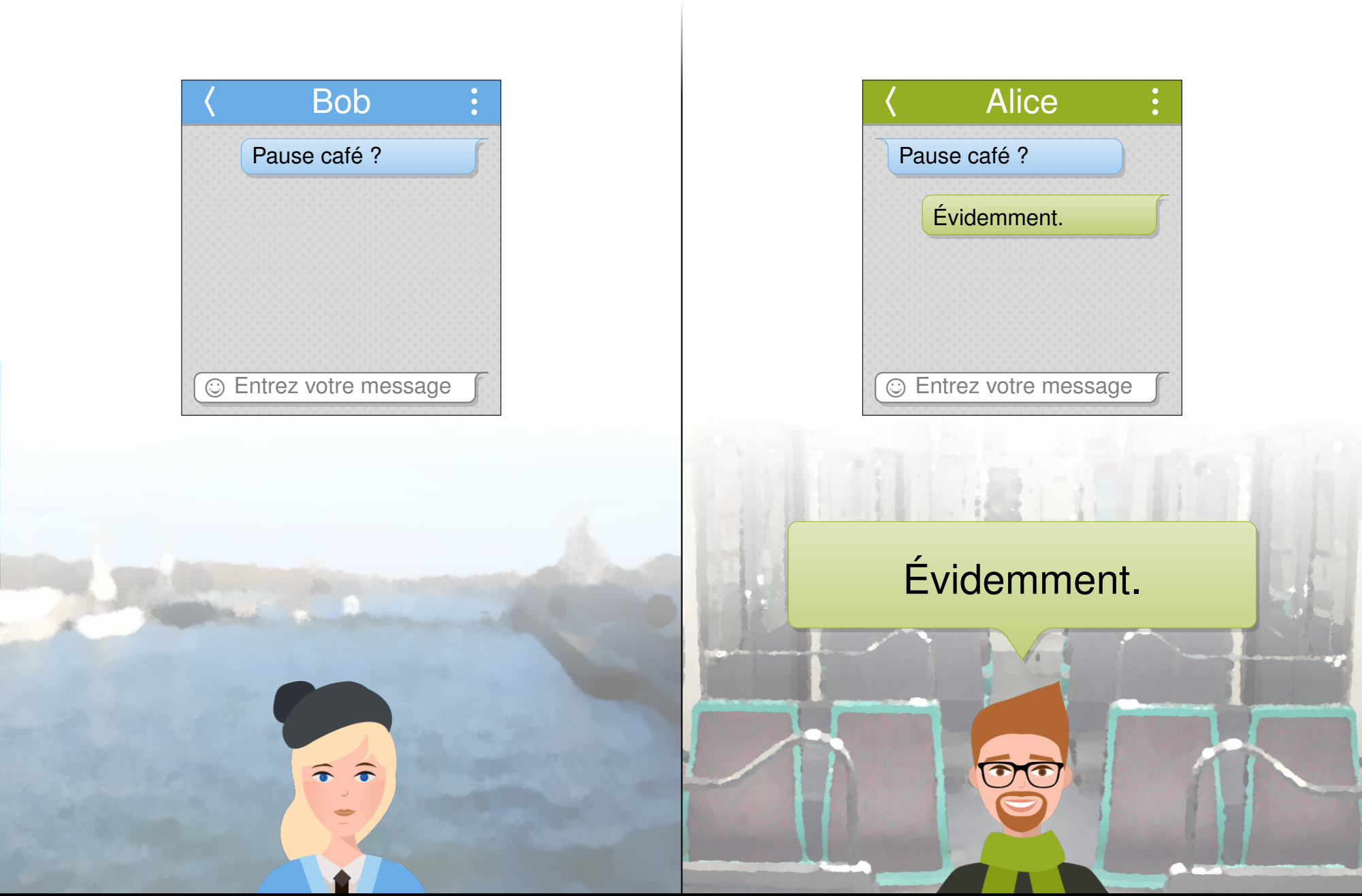
Président :	**M. Jean-Marc MENAUD**, Professeur, École des Mines de Nantes
Rapporteurs :	**M. Luc BOUGÉ**, Professeur, École Normale Supérieure de Rennes
	M^{me} Maria POTOP-BUTUCARU, Professeur, Université Pierre et Marie Curie, Paris
Examinatrice :	**M^{me} Alessia MILANI**, Maître de conférences, ENSEIRB-MATMECA, Bordeaux
Directeur de thèse :	**M. Claude JARD**, Professeur, Université de Nantes
Co-directeur de thèse :	**M. Achour MOSTÉFAOUI**, Professeur, Université de Nantes

Introduction – Motivation

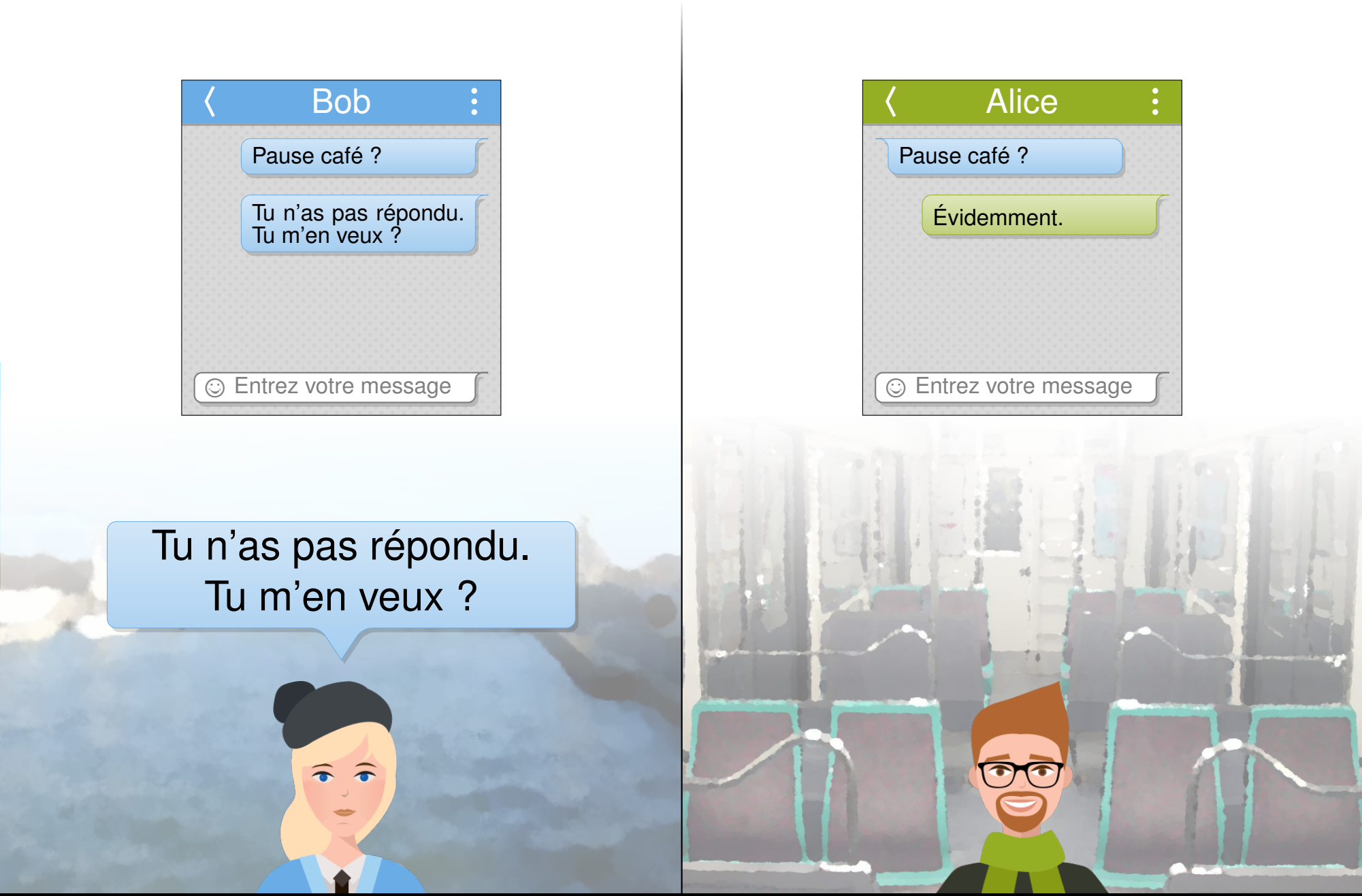
Introduction – Motivation



Introduction – Motivation

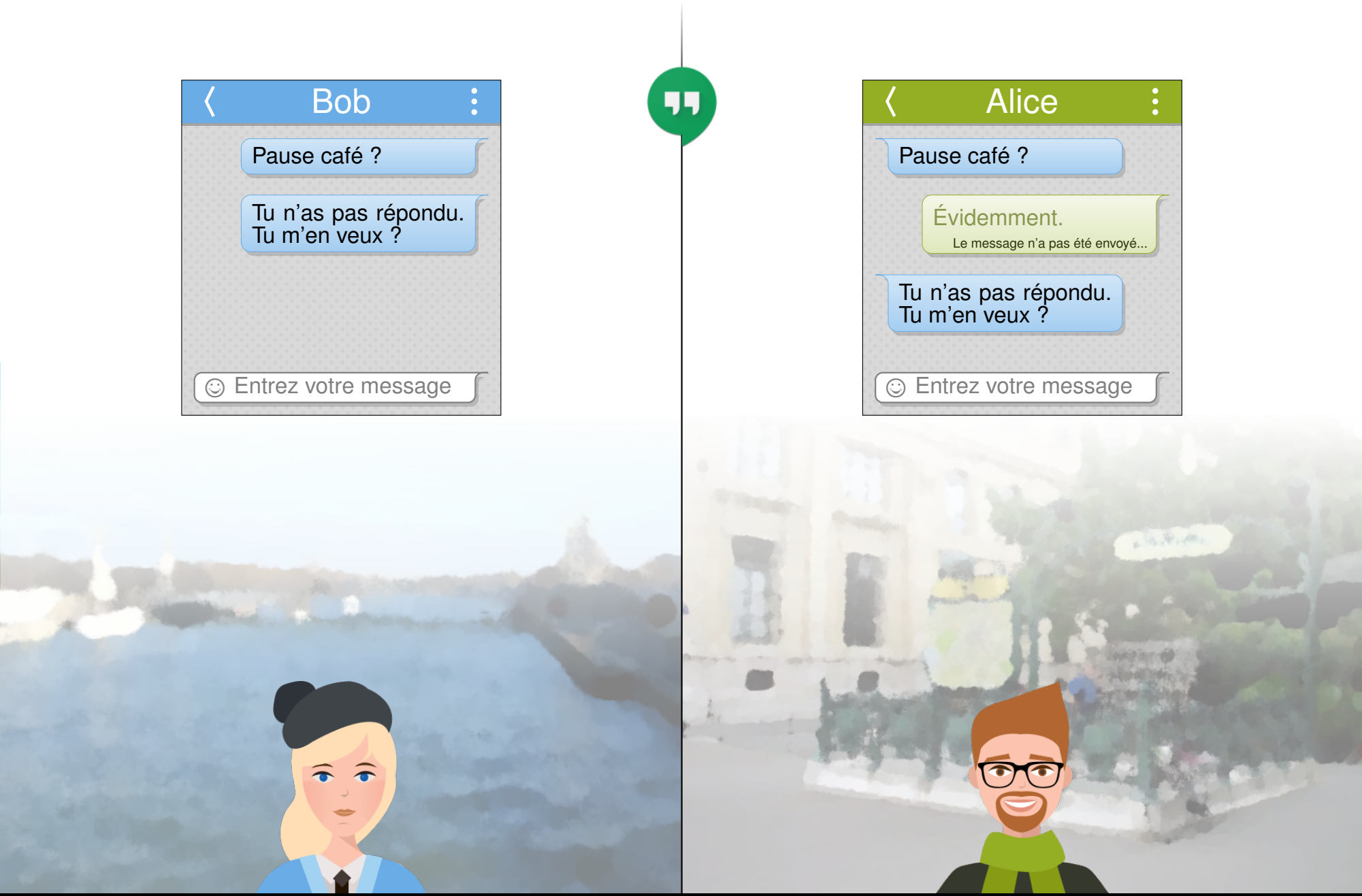


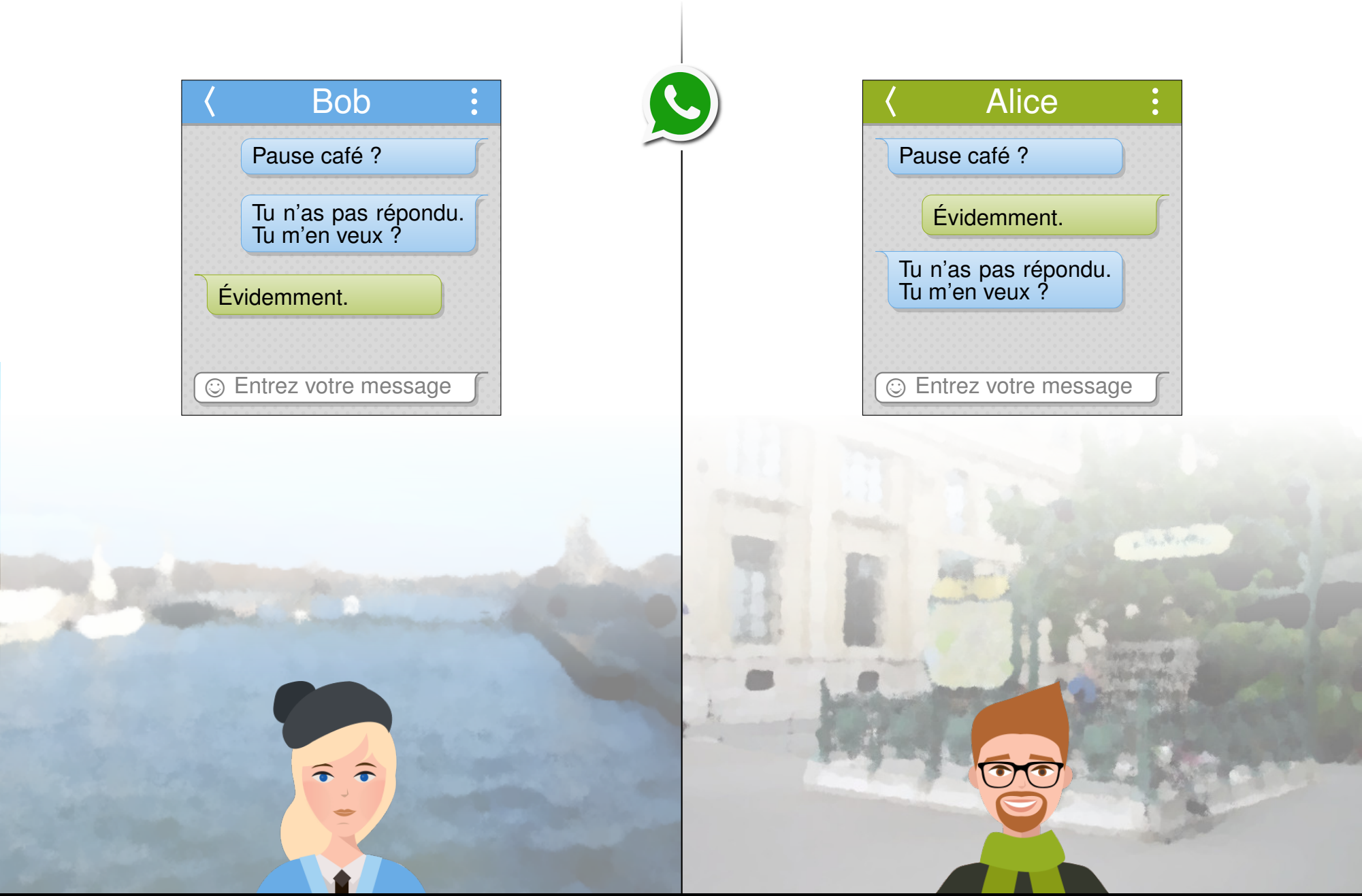
Introduction – Motivation

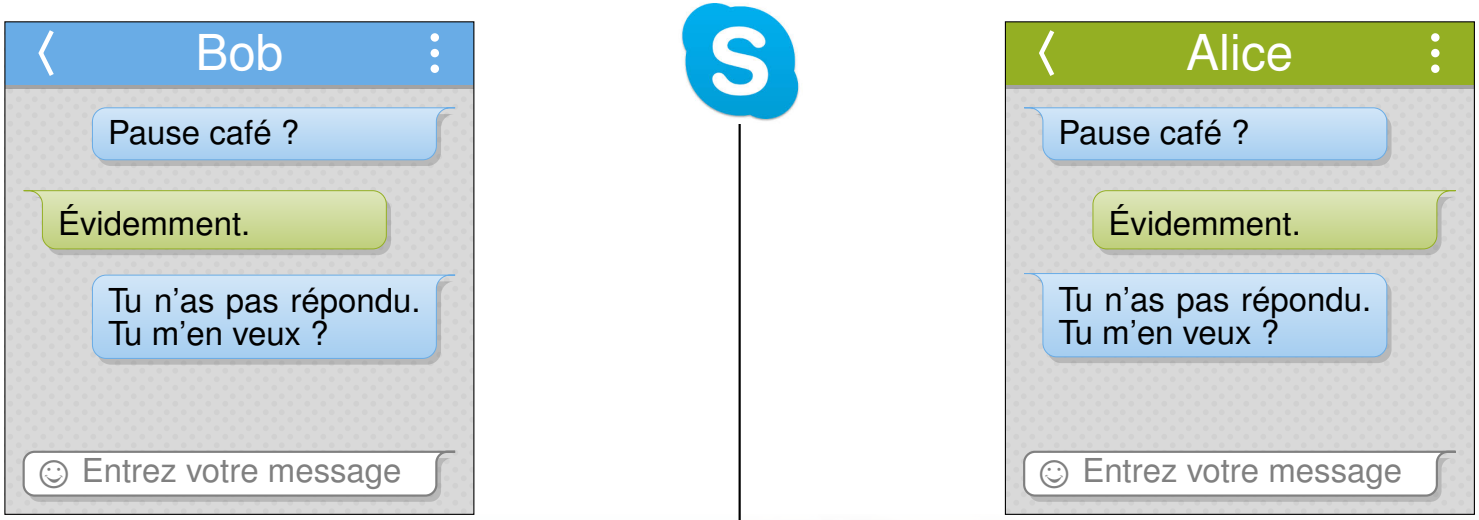


Introduction – Motivation

Introduction – Motivation : Hangouts




Bob
Pause café ?
Tu n'as pas répondu.
Tu m'en veux ?
Évidemment.
Entrez votre message
Alice
Pause café ?
Évidemment.
Tu n'as pas répondu.
Tu m'en veux ?
Entrez votre message


Bob
Pause café ?
Évidemment.
Tu n'as pas répondu.
Tu m'en veux ?
Entrez votre message
Alice
Pause café ?
Évidemment.
Tu n'as pas répondu.
Tu m'en veux ?
Entrez votre message

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*. SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.* SIGACT 2002

Concurrence

Plusieurs *processus* intéragissent avec le même *objet partagé*

- Alice et Bob
- Ordinateurs
- Fils d'exécution...

- Messagerie instantannée
- Application collaborative
- Mémoire partagée...

Systèmes répartis asynchrones à passage de messages sans-attente

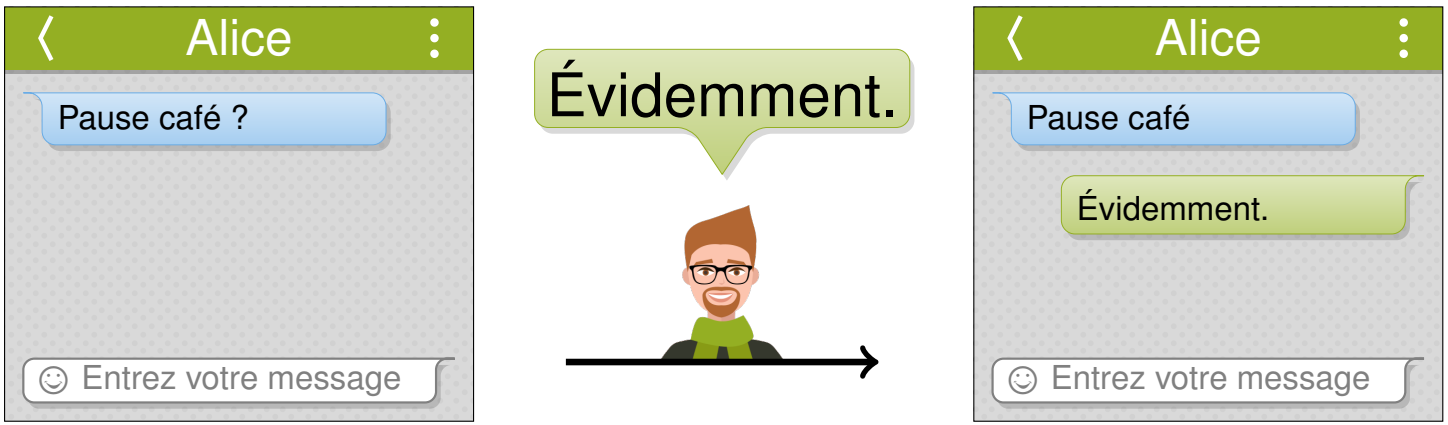
- Nombre fixe et connu de processus
- Primitives d'envoi et de réception de messages
- Délai de transmission non borné
- Retour des opérations instantané
  - Impossible d'implémenter la cohérence forte[1]

Problématique

Comment spécifier les objets partagés
d'un système sans-attente ?

[1] Gilbert, Lynch. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.* SIGACT 2002

Spécification séquentielle



Critère de cohérence

```
shared Messenger m = new Messenger();
```

- Comportement concurrent ?
- Critère de cohérence

Spécification séquentielle

```
class Messenger {
   string[] received;
   void send(string message) {
     received.length ++ ;
     received[received.length - 1] = message ;
   }
}
```

Critère de cohérence

`shared Messenger m = new Messenger();`

- Comportement concurrent ?
- Critère de cohérence

Spécification séquentielle

```
class Messenger {
   string[] received;
   void send(string message) {
     received.length ++ ;
     received[received.length - 1] = message ;
   }
}
```

Critère de cohérence

```
shared Messenger m = new Messenger();
```

- Comportement concurrent ?
- Critère de cohérence

Bases de données et mémoires transactionnelles

Transactions

Sérialisabilité

Systèmes concurrents

Convergence

CRDT

Convergence forte

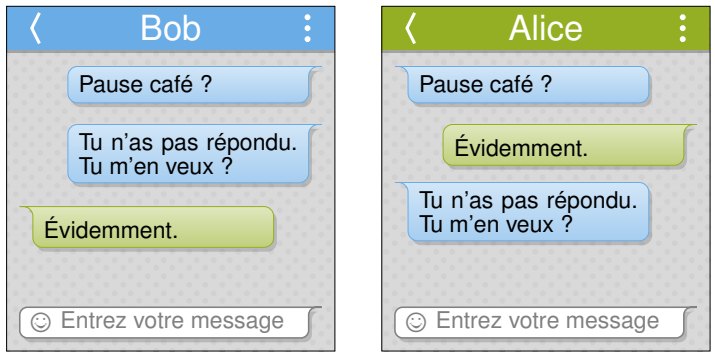
Calcul parallèle

Modèles de mémoire

PRAM

Mémoire causale

Cohérence de cache



Cohérence séquentielle

Linéarisabilité

Consensus

Algorithmique du réparti

Bases de données et mémoires transactionnelles

Transactions

Sérialisabilité

Systèmes concurrents

Convergence

CRDT

Convergence forte

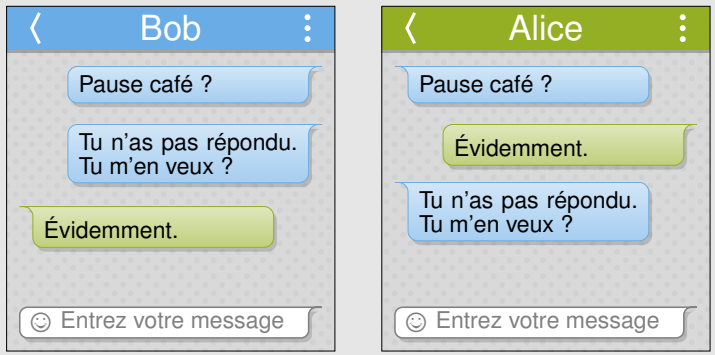
Calcul parallèle

Modèles de mémoire

PRAM

Mémoire causale

Cohérence de cache



Cohérence séquentielle

Linéarisabilité

Consensus

Algorithmique du réparti

Bases de données et mémoires transactionnelles

Transactions

Sérialisabilité

Systèmes concurrents

Calcul parallèle

Modèles de mémoire

PRAM

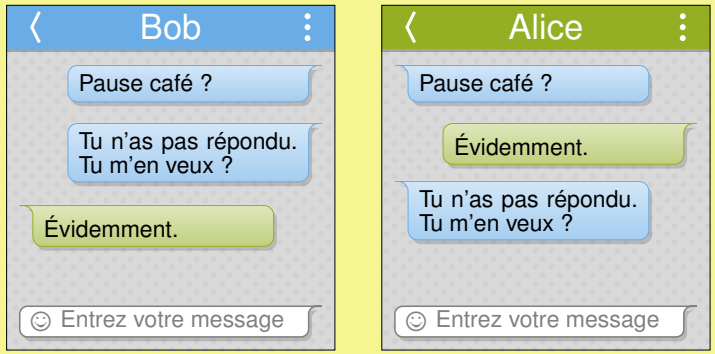
Mémoire causale

Cohérence de cache

Convergence

CRDT

Convergence forte



Cohérence séquentielle

Linéarisabilité

Consensus

Algorithmique du réparti

Bases de données et mémoires transactionnelles

Transactions

Sérialisabilité

Systèmes concurrents

Convergence

CRDT

Convergence forte

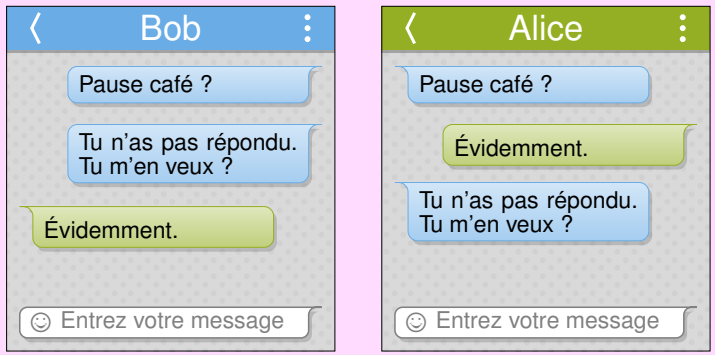
Calcul parallèle

Modèles de mémoire

PRAM

Mémoire causale

Cohérence de cache



Cohérence séquentielle

Linéarisabilité

Consensus

Algorithmique du réparti

Bases de données et mémoires transactionnelles

Transactions

Sérialisabilité

Systèmes concurrents

Convergence

CRDT

Convergence forte

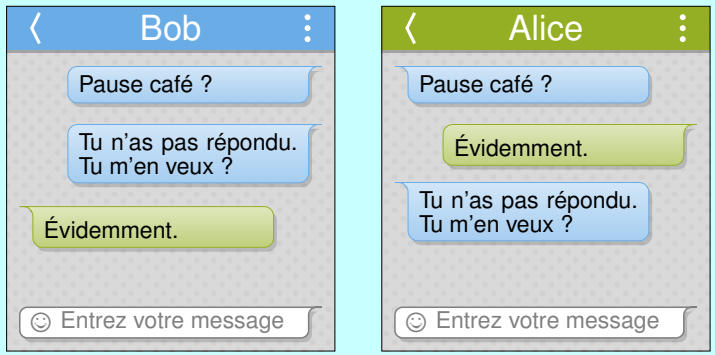
Calcul parallèle

Modèles de mémoire

PRAM

Mémoire causale

Cohérence de cache



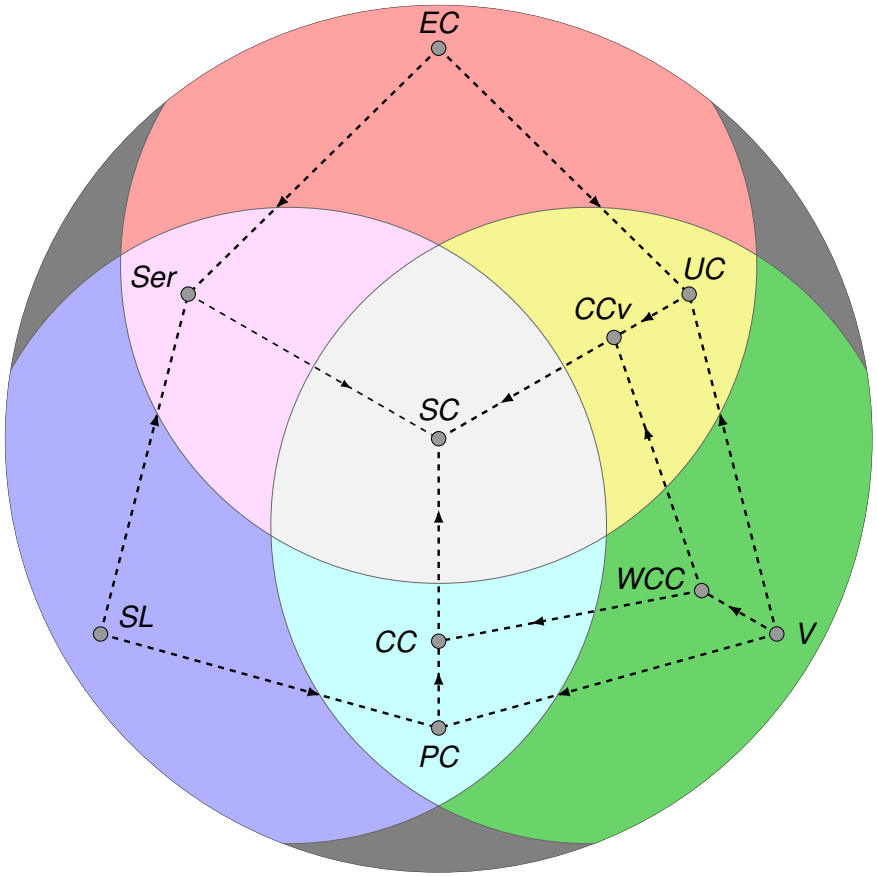
Cohérence séquentielle

Linéarisabilité

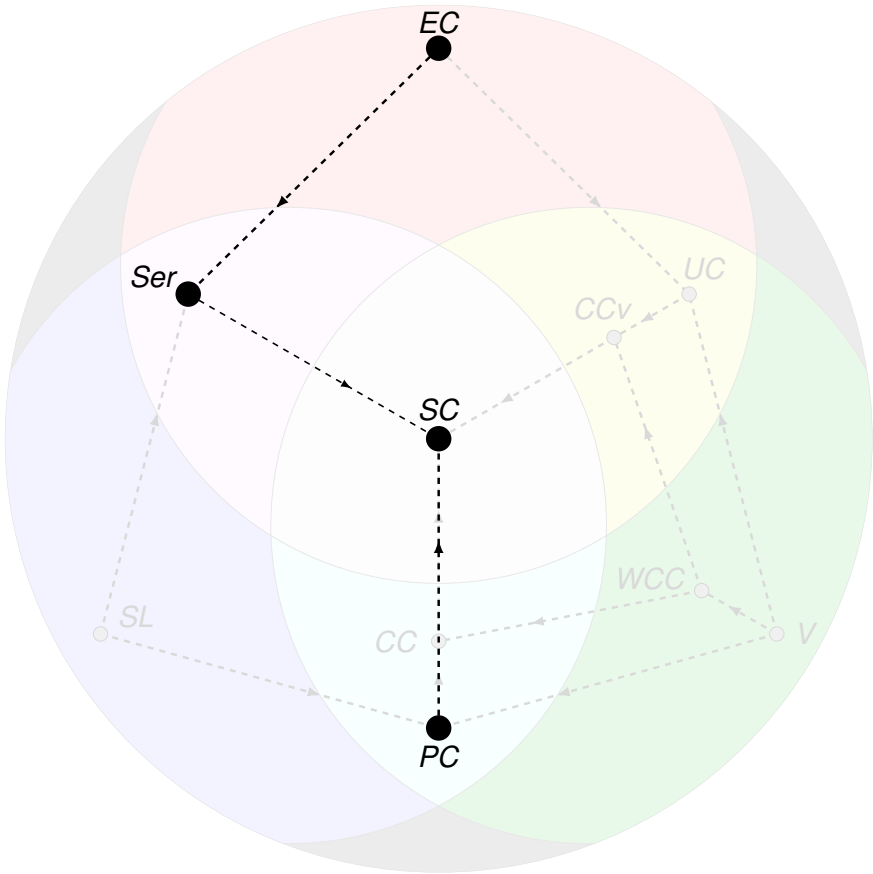
Consensus

Algorithmique du réparti

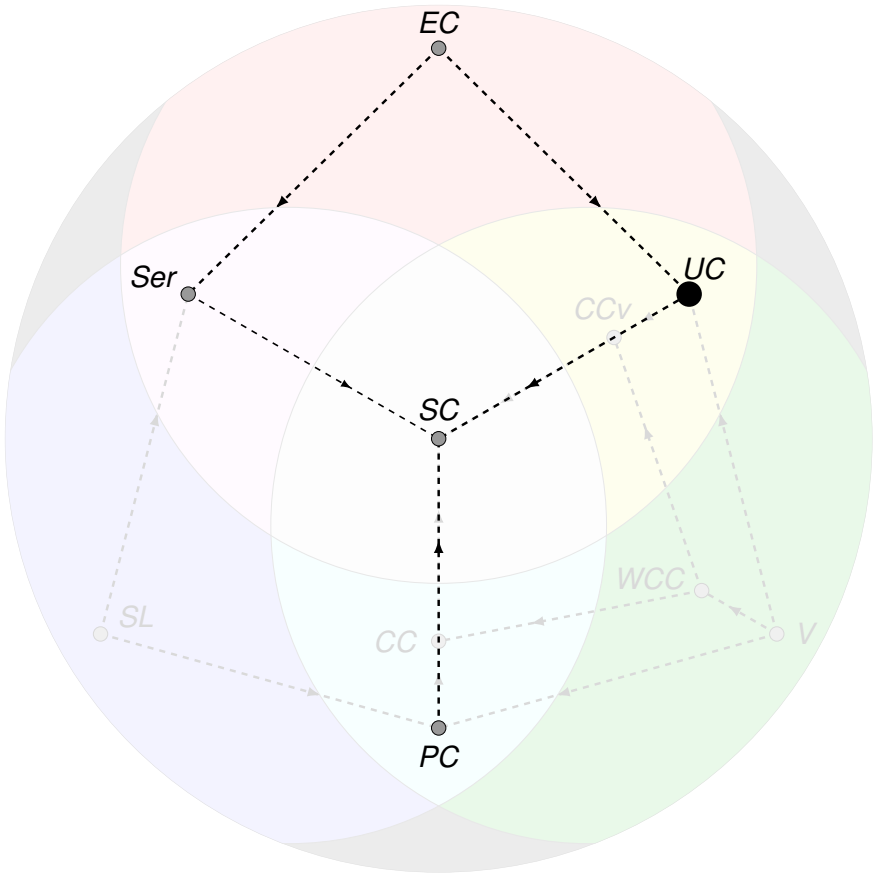
Introduction – Plan



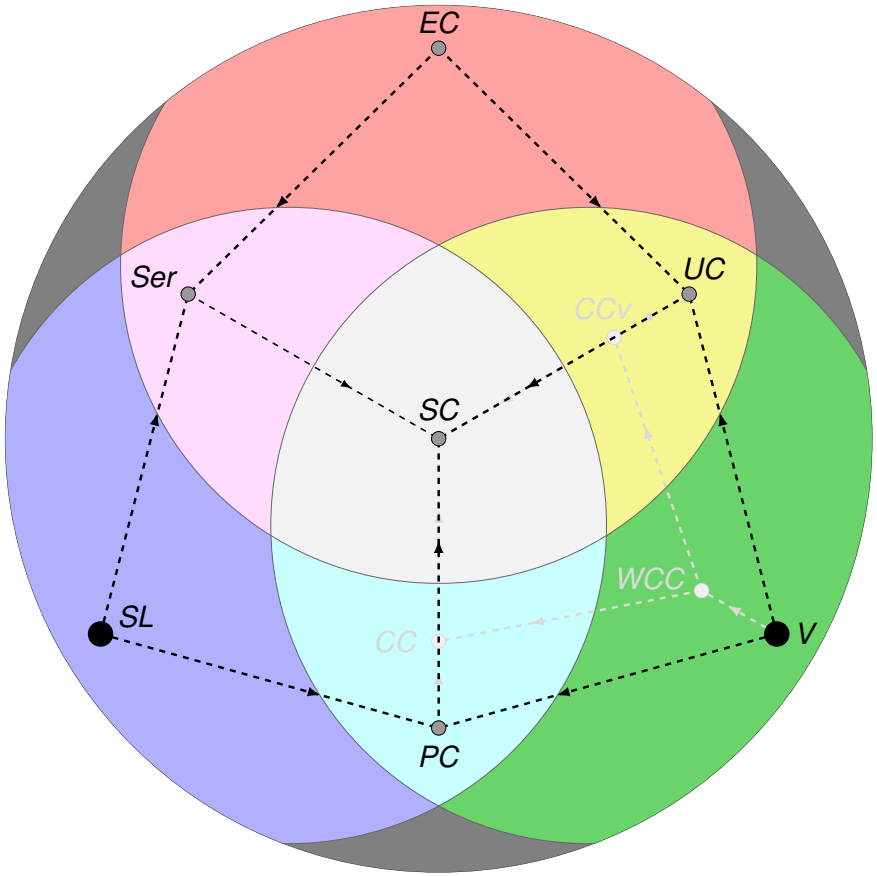
Introduction – Plan



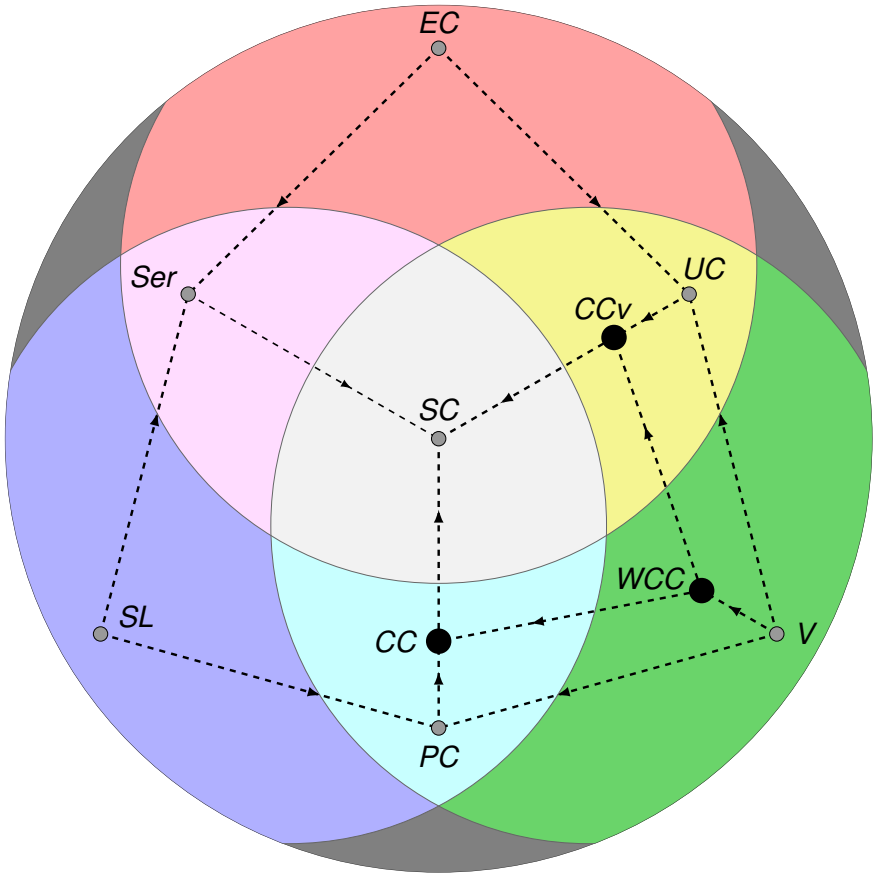
Introduction – Plan



Introduction – Plan

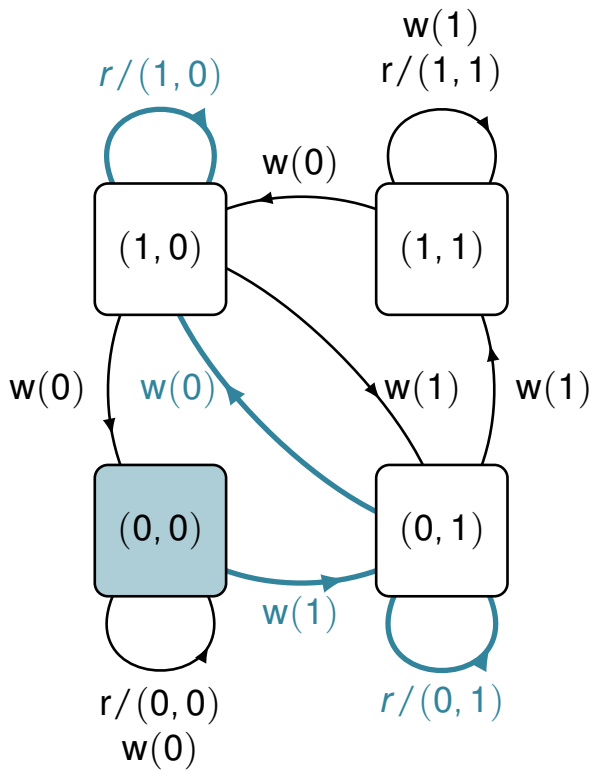


Introduction – Plan



1. Modèle – *Flux fenêtré de taille k*

Type de données abstrait (ADT)



Deux types d'opération

Écritures

- $v \in \mathbb{N}$: message
- $\mathrm{w}(v)$: envoi de v

Lectures

- $\mathrm{r}/(v_1, \ldots, v_k)$: rafraîchissement
- k dernières valeurs écrites ordonnées
- k : taille de l'écran

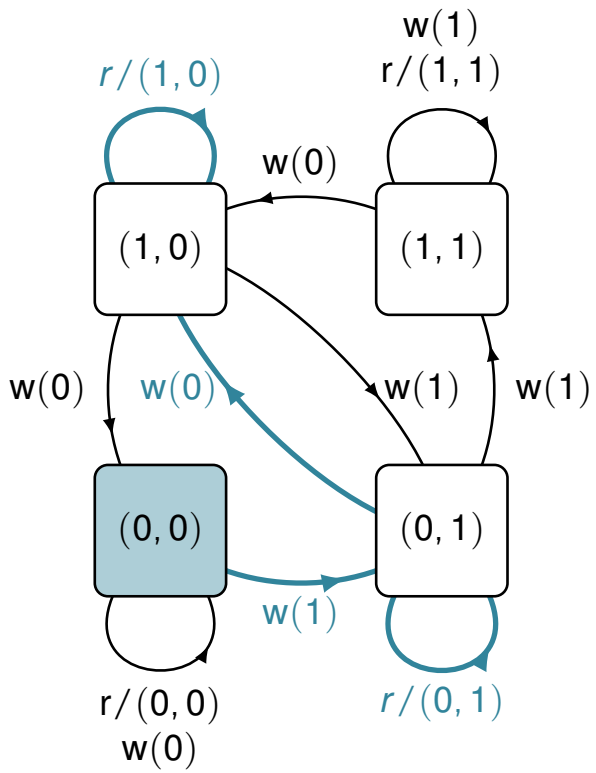
Cas général

Partie lecture : valeur de retour

Partie écriture : changement d'état

1. Modèle – *Flux fenêtré de taille k*

Type de données abstrait (ADT)



Deux types d'opération

Écritures

- $v \in \mathbb{N}$: message
- $\mathrm{w}(v)$: envoi de v

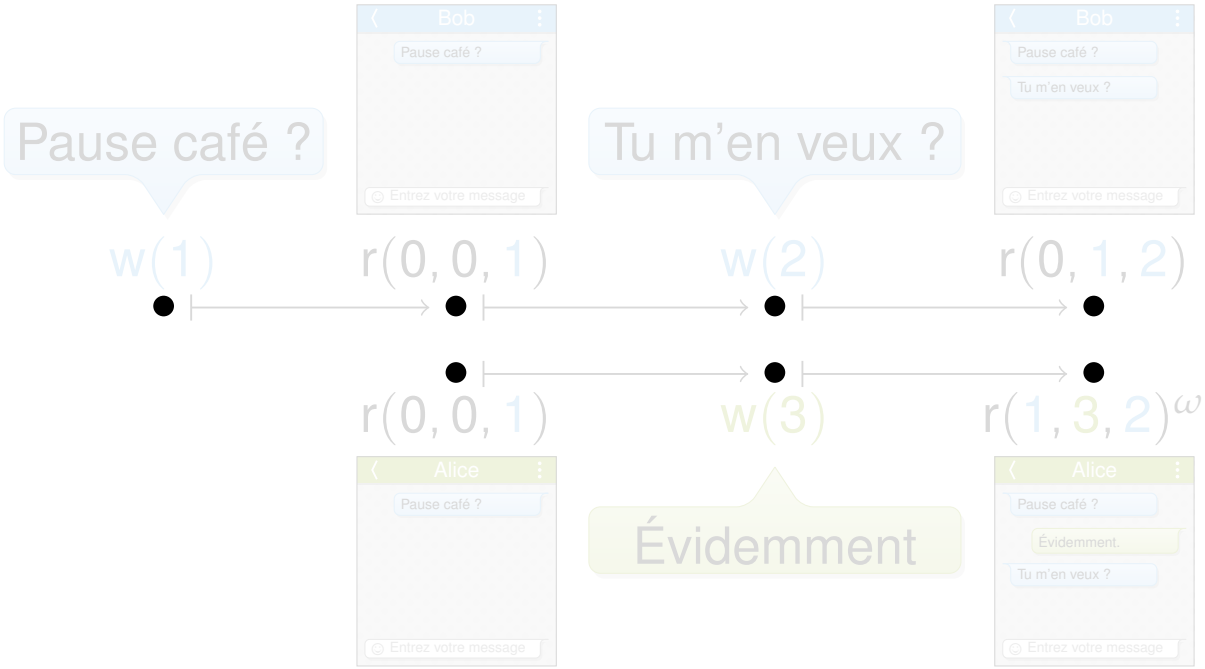
Lectures

- $\mathrm{r}/(v_1, \ldots, v_k)$: rafraîchissement
- k dernières valeurs écrites ordonnées
- k : taille de l'écran

Spécification séquentielle

- Ensemble des chemins depuis l'état initial
- Exemple : $\mathrm{w}(1) \cdot \mathrm{r}/(0,1) \cdot \mathrm{w}(0) \cdot \mathrm{r}/(1,0)^\omega$

1. Modèle – *Histoire concurrente*

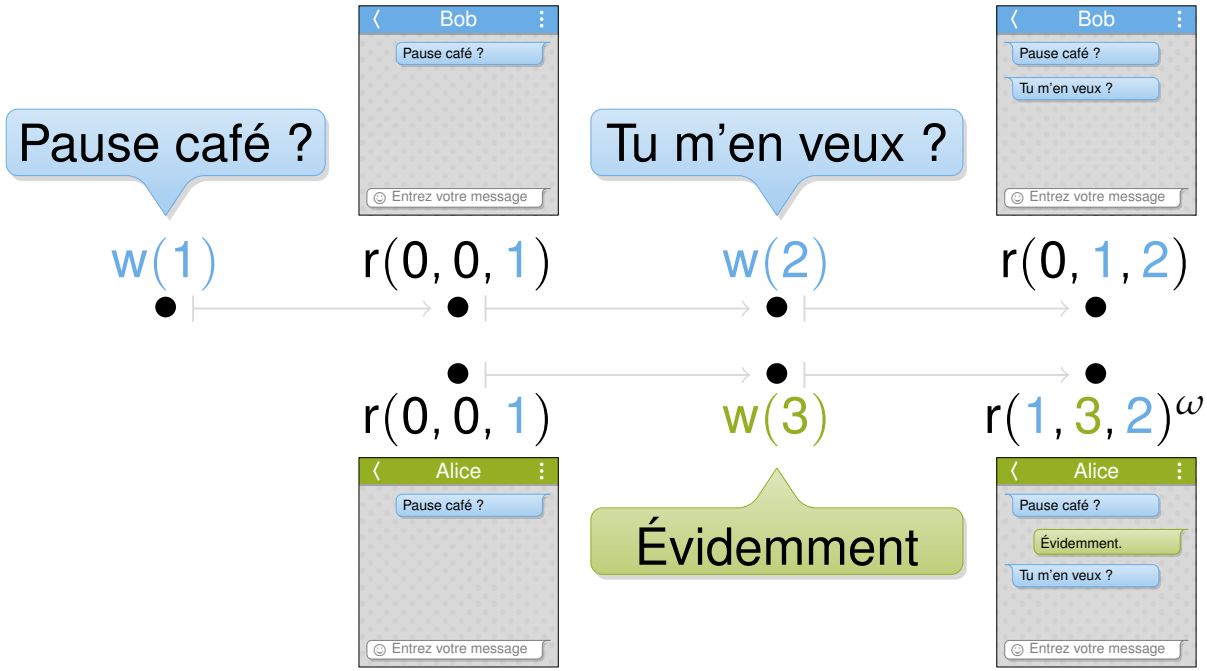


Modélisation d'une exécution

- Événements
- Opérations
- Ordre de processus

Cas général

- Création de processus pendant l'exécution
- Préemption possible

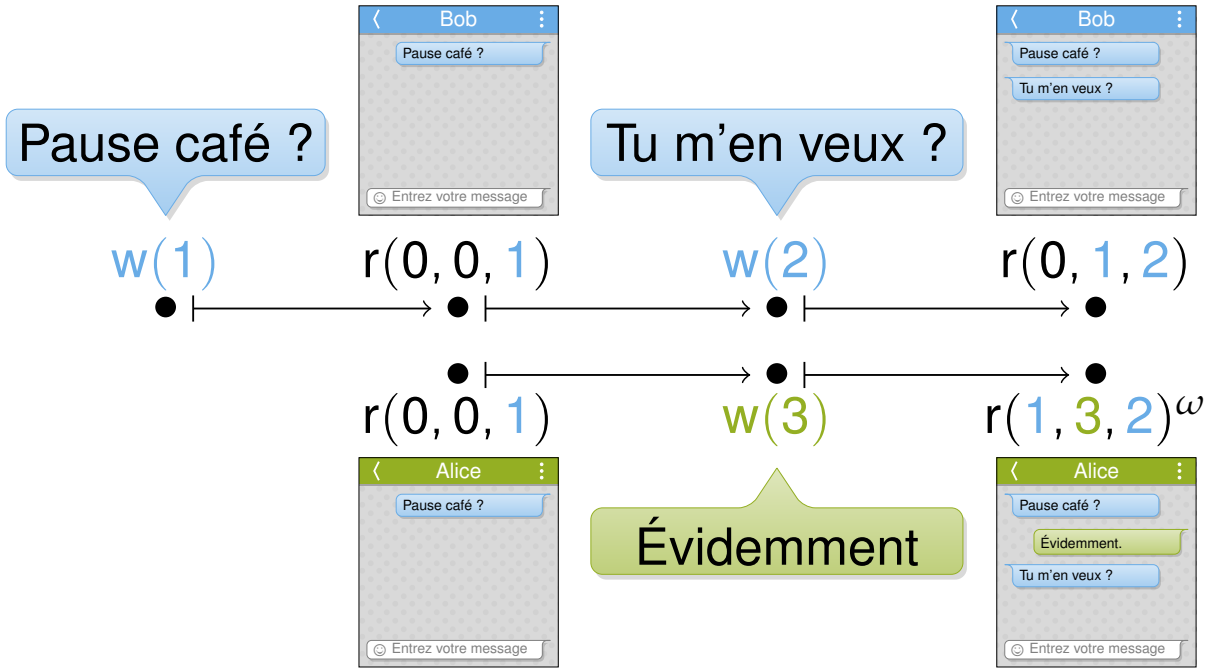


Modélisation d'une exécution

- Événements
- Opérations
- Ordre de processus

Cas général

- Création de processus pendant l'exécution
- Préemption possible

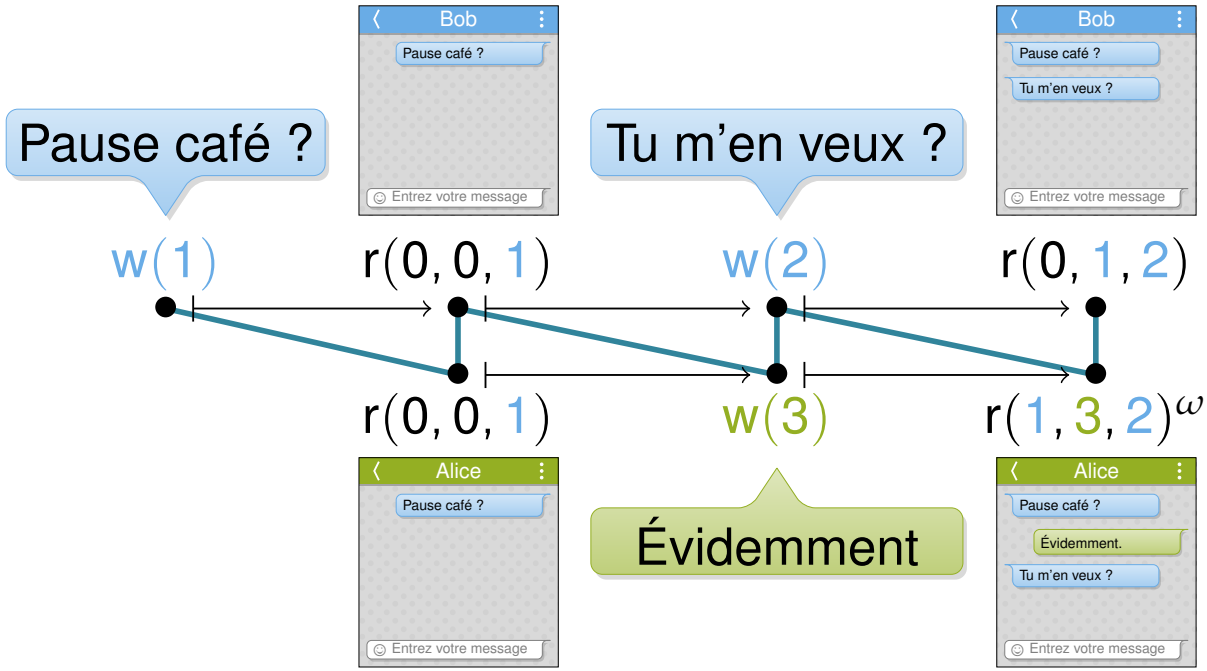


Modélisation d'une exécution

- Événements
- Opérations
- Ordre de processus

Cas général

- Création de processus pendant l'exécution
- Préemption possible



Linéarisation

- Séquence d'opérations
- Ordre total sur les événements
- Respecte l'ordre de processus

Définition

- C : ADT $\rightarrow$ ensemble d'histoires concurrentes

Structure de treillis

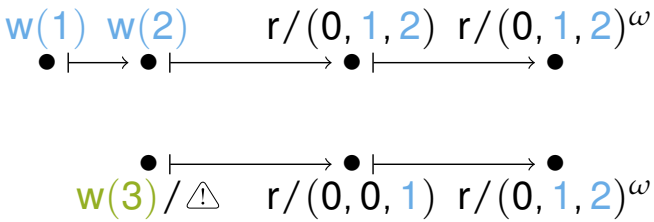
- C_1 plus fort que C_2 $\qquad \forall T, C_1(T) \subset C_2(T)$
  - C_1 admet moins d'histoires que C_2
- Conjonction $\qquad (C_1 + C_2)(T) = C_1(T) \cap C_2(T)$
  - $C_1 + C_2$ plus fort que C_1 et que C_2

Exemple : la cohérence séquentielle

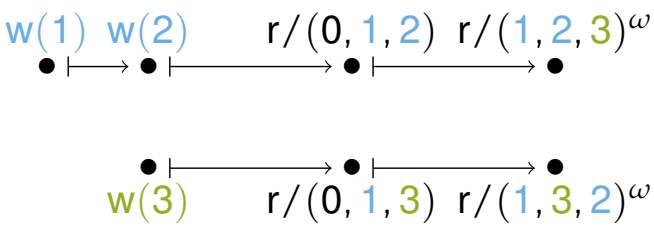
$H \in SC(T)$: il existe une linéarisation de H admise par T

1. Modèle – Trois services de messagerie

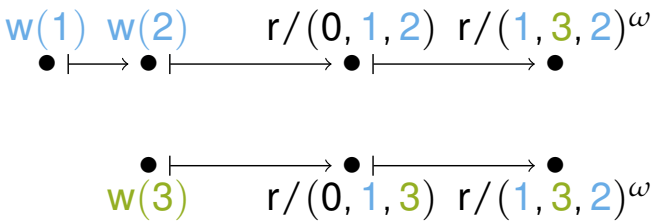
Hangouts



WhatsApp



Skype



Sérialisabilité (Ser)[1]

- Écritures *avortées* : erreur ⚠
- Autres événements dans une linéarisation admise par T

Cohérence pipeline (PC)[2]

Une linéarisation admise par T par processus contient

- Toutes les écritures de H
- Les lectures du processus

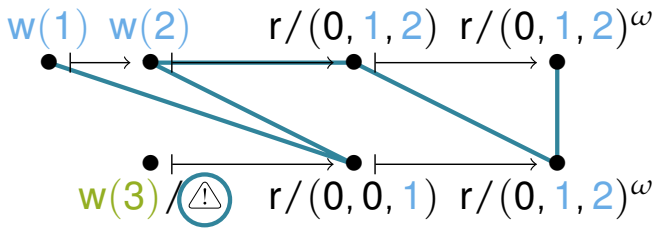
Convergence (EC)[3]

- À terme, toutes les lectures sont faites dans le même état

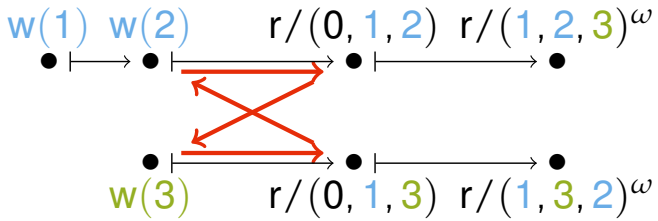
[1] Papadimitriou. *The serializability of concurrent database updates*. 1979
[2] Lipton, Sandberg. *PRAM: A Scalable Shared Memory*. 1988
[3] Vogels. *Eventually consistent*. 2009

1. Modèle – Trois services de messagerie

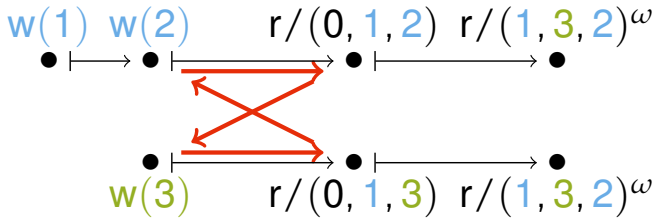
Hangouts



WhatsApp



Skype



Sérialisabilité (*Ser*)[1]

- Écritures *avortées* : erreur ⚠
- Autres événements dans une linéarisation admise par T

Cohérence pipeline (*PC*)[2]

Une linéarisation admise par T par processus contient

- Toutes les écritures de H
- Les lectures du processus

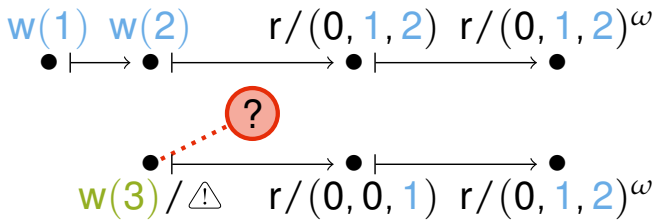
Convergence (*EC*)[3]

- À terme, toutes les lectures sont faites dans le même état

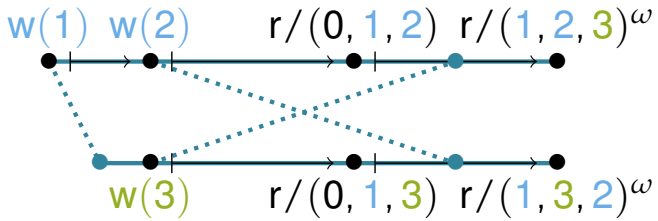
[1] Papadimitriou. *The serializability of concurrent database updates*. 1979
[2] Lipton, Sandberg. *PRAM: A Scalable Shared Memory*. 1988
[3] Vogels. *Eventually consistent*. 2009

1. Modèle – Trois services de messagerie

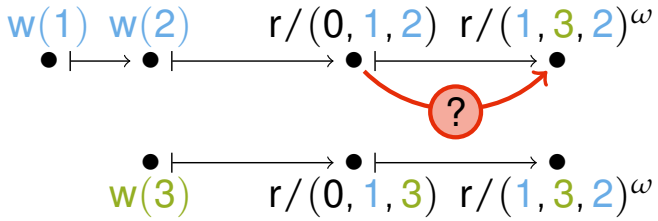
Hangouts



WhatsApp



Skype



Sérialisabilité (*Ser*)[1]

- Écritures *avortées* : erreur ⚠
- Autres événements dans une linéarisation admise par T

Cohérence pipeline (*PC*)[2]

Une linéarisation admise par T par processus contient

- Toutes les écritures de H
- Les lectures du processus

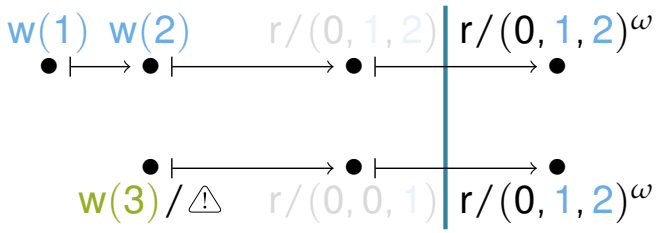
Convergence (*EC*)[3]

- À terme, toutes les lectures sont faites dans le même état

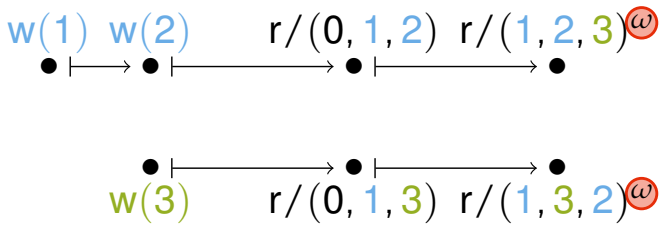
[1] Papadimitriou. *The serializability of concurrent database updates*. 1979
[2] Lipton, Sandberg. *PRAM: A Scalable Shared Memory.* 1988
[3] Vogels. *Eventually consistent*. 2009

1. Modèle – Trois services de messagerie

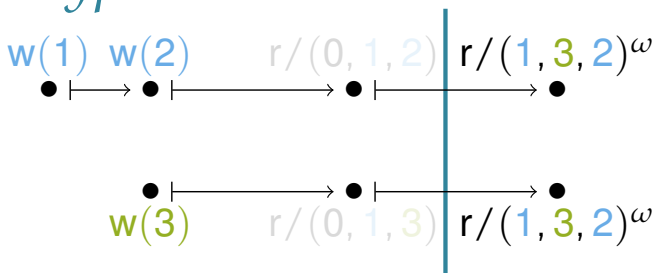
Hangouts



WhatsApp



Skype



Sérialisabilité (*Ser*)[1]

- Écritures *avortées* : erreur ⚠
- Autres événements dans une linéarisation admise par T

Cohérence pipeline (*PC*)[2]

Une linéarisation admise par T par processus contient

- Toutes les écritures de H
- Les lectures du processus

Convergence (*EC*)[3]

- À terme, toutes les lectures sont faites dans le même état

[1] Papadimitriou. *The serializability of concurrent database updates*. 1979
[2] Lipton, Sandberg. *PRAM: A Scalable Shared Memory.* 1988
[3] Vogels. *Eventually consistent*. 2009

2. Cohérence d'écritures – Motivation

Convergence forte[1]

Convergence :	à terme	$\Rightarrow$ même état
Convergence forte :	mêmes opérations *visibles*	$\Rightarrow$ même état

Spécifications concurrentes[2]

Opérations visibles $\rightarrow$ État

Limites

- Absence de modularité dans la spécification
- Liens forts avec l'implémentation
- Restreint à un seul type d'implémentation

[1] Shapiro, Preguiça, Baquero, Zawirski. *Conflict-free replicated data types*. SSS, 2011
[2] Burckhardt, Gotsman, Yang, Zawirski. *Replicated data types: specification, verification, optimality*. POPL, 2014

Convergence forte[1]

Convergence :	à terme	$\Rightarrow$ même état
Convergence forte :	mêmes opérations *visibles*	$\Rightarrow$ même état

Spécifications concurrentes[2]

Opérations visibles $\rightarrow$ État

Limites

- Absence de modularité dans la spécification
- Liens forts avec l'implémentation
- Restreint à un seul type d'implémentation

[1] Shapiro, Preguiça, Baquero, Zawirski. *Conflict-free replicated data types*. SSS, 2011
[2] Burckhardt, Gotsman, Yang, Zawirski. *Replicated data types: specification, verification, optimality*. POPL, 2014

Convergence forte[1]

Convergence :	à terme	$\Rightarrow$ même état
Convergence forte :	mêmes opérations *visibles*	$\Rightarrow$ même état

Spécifications concurrentes[2]

Opérations visibles $\rightarrow$ État

Limites

- Absence de modularité dans la spécification
- Liens forts avec l'implémentation
- Restreint à un seul type d'implémentation

[1] Shapiro, Preguiça, Baquero, Zawirski. *Conflict-free replicated data types*. SSS, 2011
[2] Burckhardt, Gotsman, Yang, Zawirski. *Replicated data types: specification, verification, optimality*. POPL, 2014

Convergence forte[1]

Convergence :	à terme	$\Rightarrow$ même état
Convergence forte :	mêmes opérations *visibles*	$\Rightarrow$ même état

Spécifications concurrentes[2]

$$\underbrace{\text{Opérations visibles}}_{\text{messages reçus}} \rightarrow \text{État}$$

Limites

- Absence de modularité dans la spécification
- Liens forts avec l'implémentation
- Restreint à un seul type d'implémentation

[1] Shapiro, Preguiça, Baquero, Zawirski. *Conflict-free replicated data types*. SSS, 2011
[2] Burckhardt, Gotsman, Yang, Zawirski. *Replicated data types: specification, verification, optimality*. POPL, 2014

Convergence forte[1]

Convergence :	à terme	$\Rightarrow$ même état
Convergence forte :	mêmes opérations *visibles*	$\Rightarrow$ même état

Spécifications concurrentes[2]

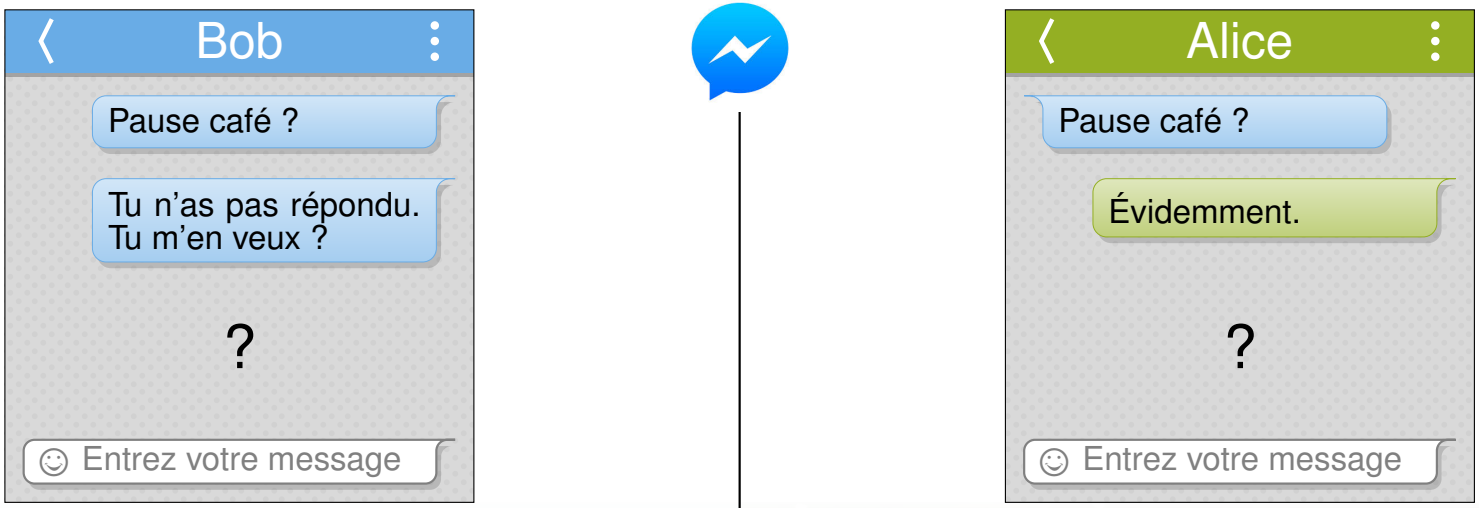
$$\underbrace{\text{Opérations visibles}}_{\text{messages reçus}} \rightarrow \text{État}$$

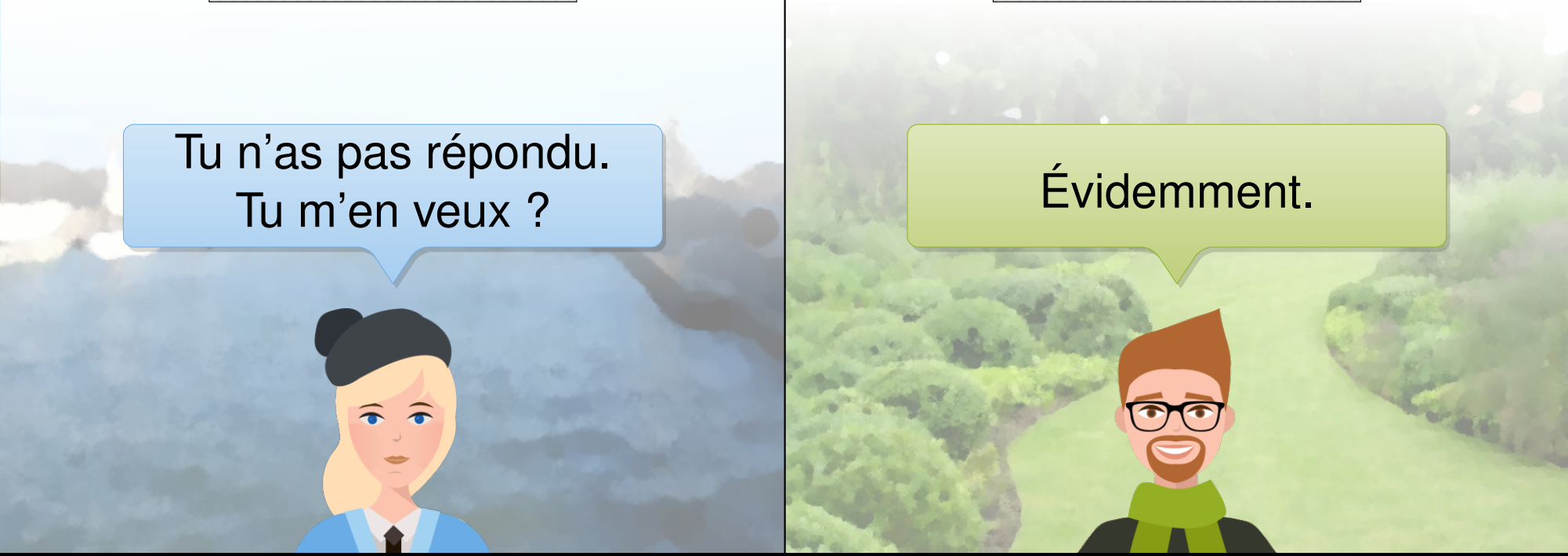
Limites

- Absence de modularité dans la spécification
- Liens forts avec l'implémentation
- Restreint à un seul type d'implémentation

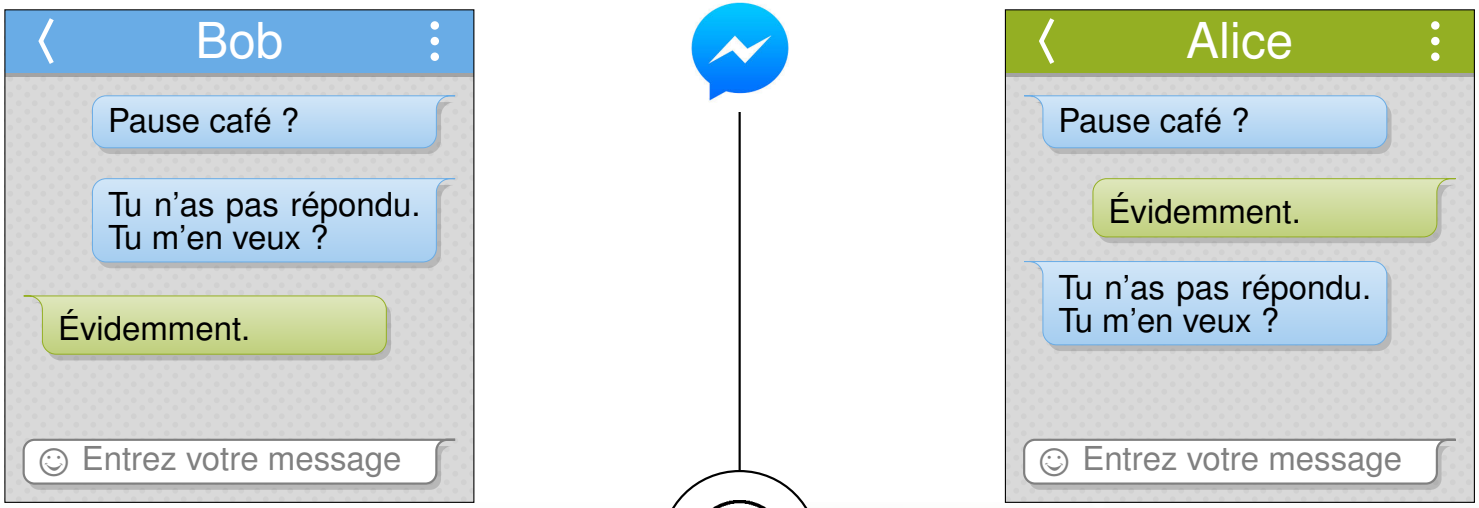
[1] Shapiro, Preguiça, Baquero, Zawirski. *Conflict-free replicated data types*. SSS, 2011
[2] Burckhardt, Gotsman, Yang, Zawirski. *Replicated data types: specification, verification, optimality*. POPL, 2014

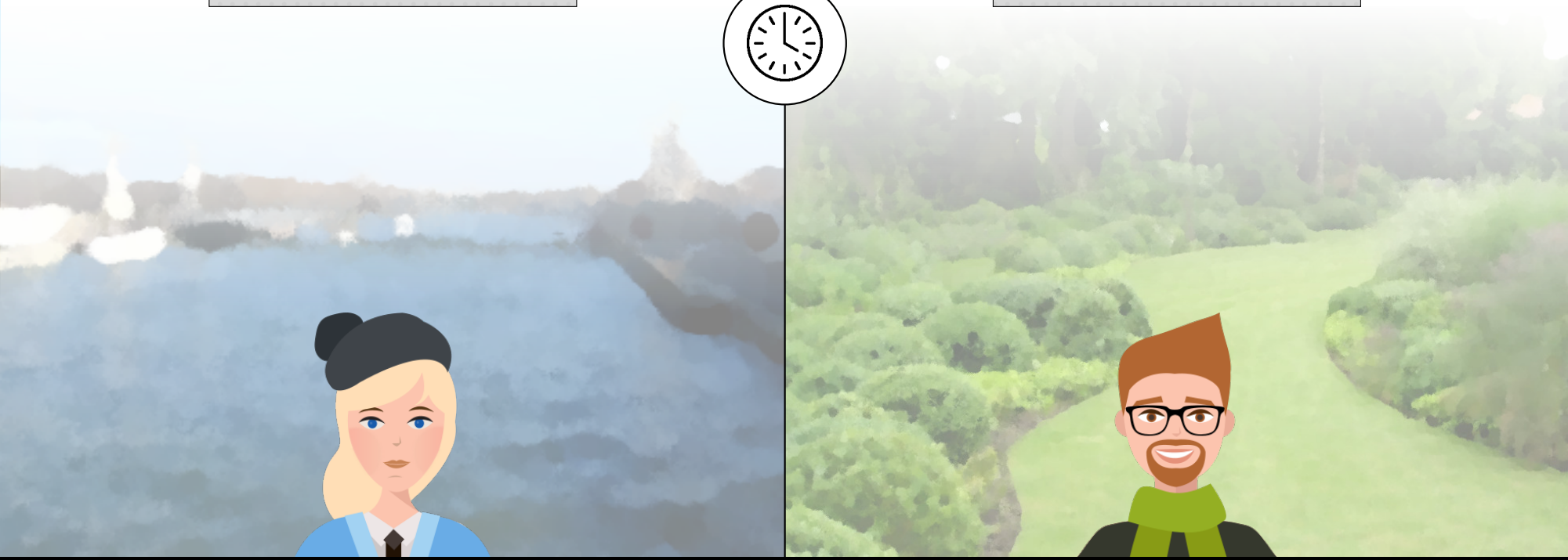
2. Cohérence d'écritures – Facebook Messenger



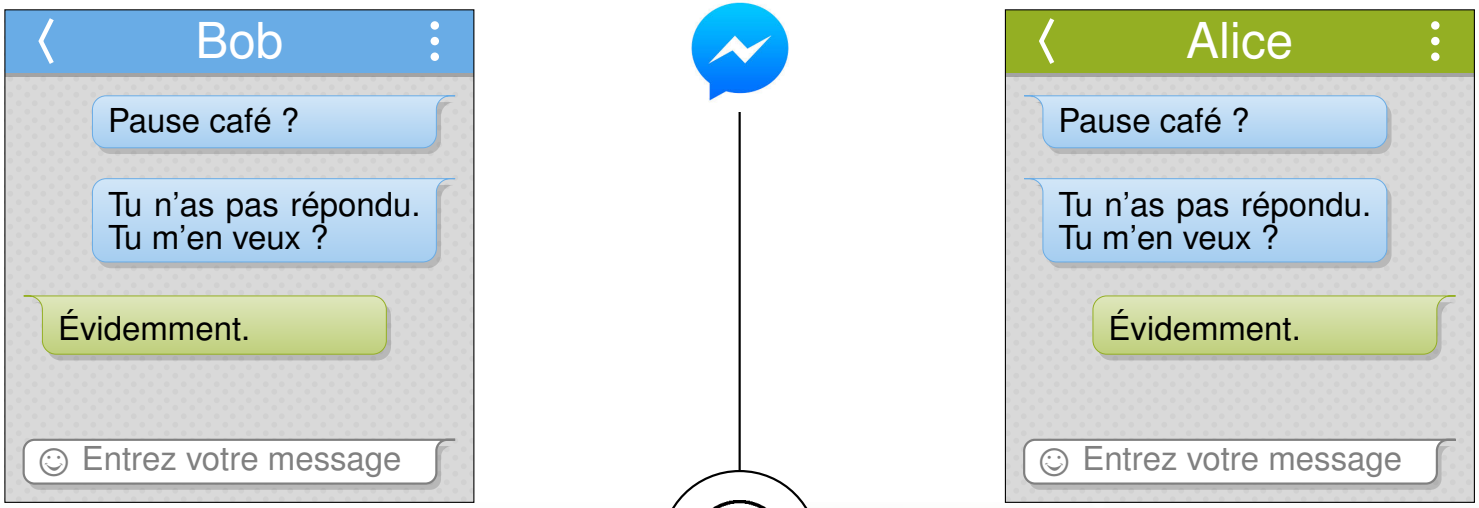


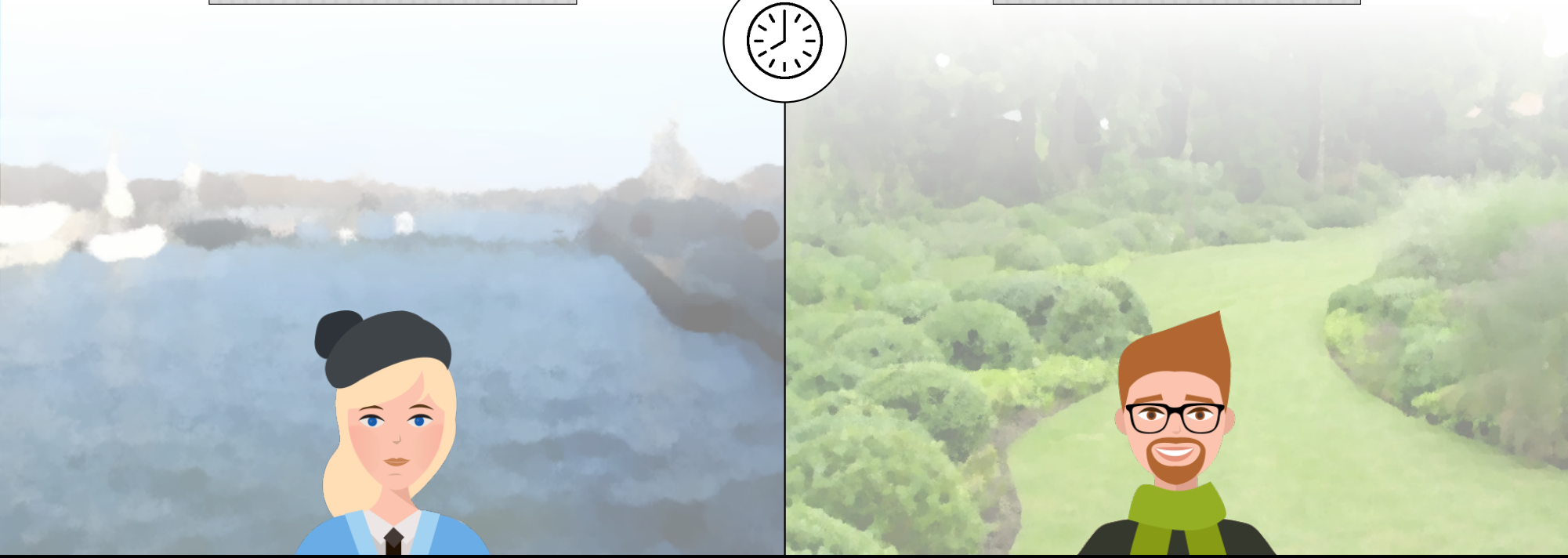
2. Cohérence d'écritures – Facebook Messenger

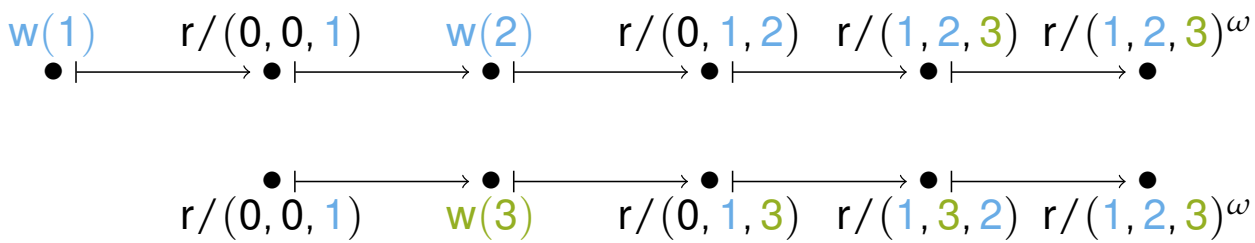




2. Cohérence d'écritures – Facebook Messenger



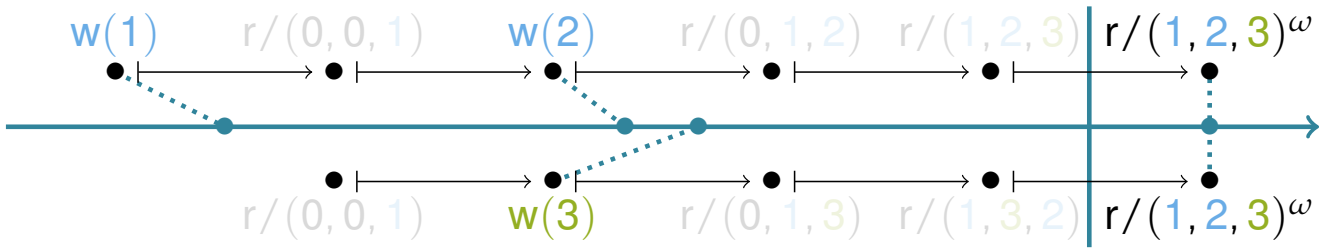




H ∈ *UC*(*T*) *si*

- Contient une infinité d'écritures, ou
- En dehors d'un nombre fini de lectures, *H* ∈ *SC*(*T*)

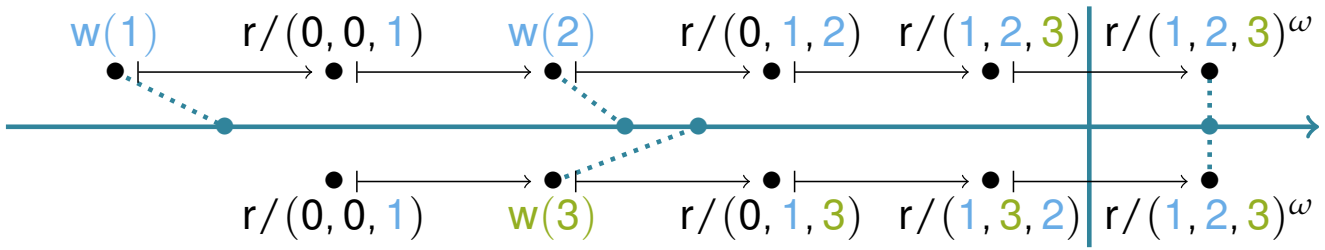
2. Cohérence d'écritures – Définition



H ∈ *UC*(*T*) *si*

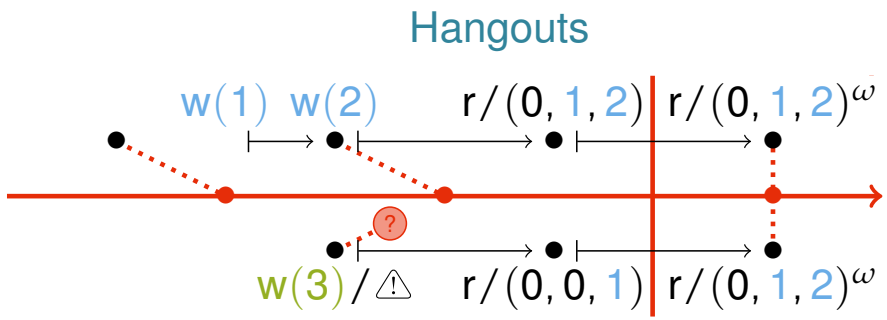
- Contient une infinité d'écritures, ou
- En dehors d'un nombre fini de lectures, *H* ∈ *SC*(*T*)

2. Cohérence d'écritures – Définition



$H \in UC(T)$ *si*

- Contient une infinité d'écritures, ou
- En dehors d'un nombre fini de lectures, $H \in SC(T)$

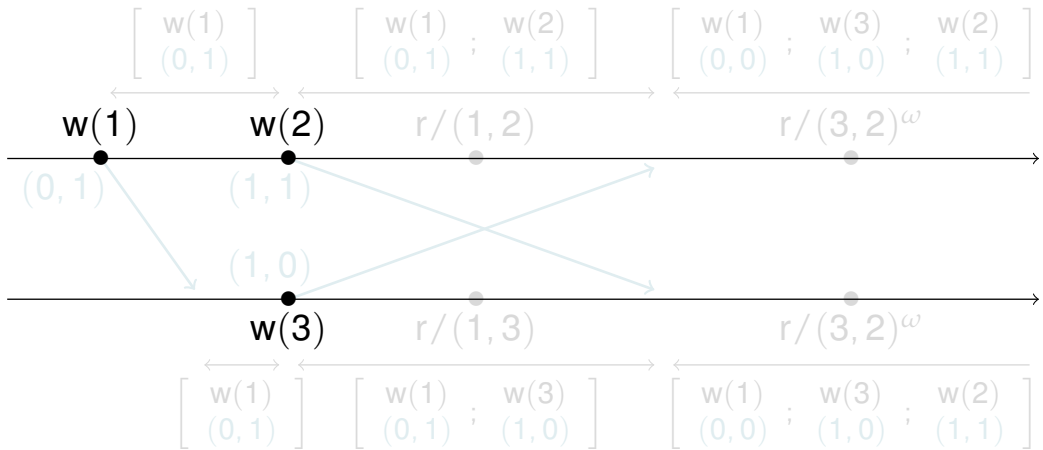


2. Cohérence d'écritures – Implémentation

Idée

- Construction d'un ordre total *a priori*
- Écriture : envoi d'un message
- Lecture : exécute toute l'histoire

Exemple



Autres idées envisagées

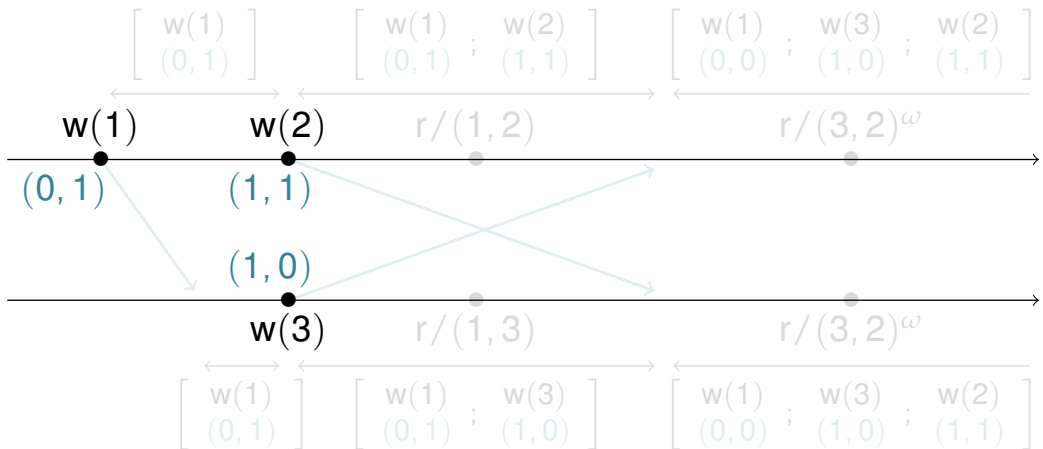
- Exécution locale des opérations à la réception du message
- Convergence asynchrone en envoyant d'autres messages

2. Cohérence d'écritures – Implémentation

Idée

- Construction d'un ordre total *a priori*
- Écriture : envoi d'un message
- Lecture : exécute toute l'histoire

Exemple



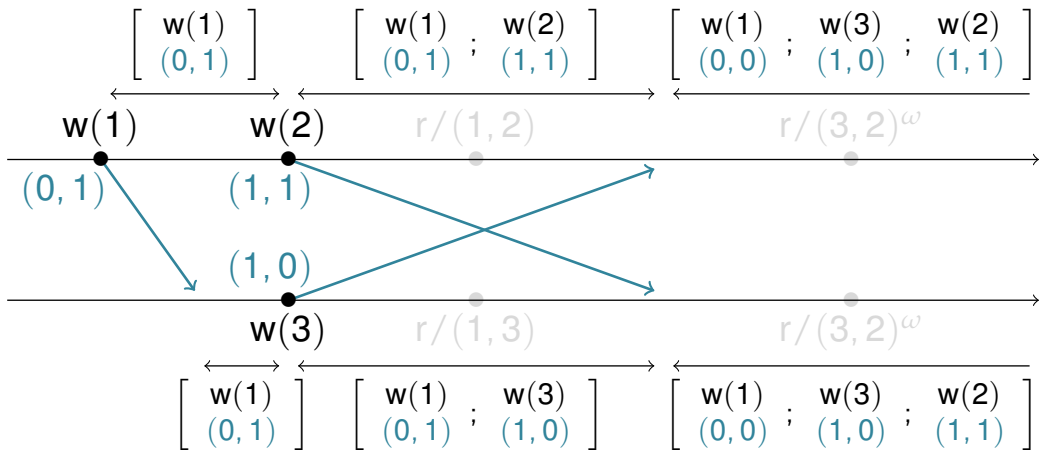
Autres idées envisagées

- Exécution locale des opérations à la réception du message
- Convergence asynchrone en envoyant d'autres messages

Idée

- Construction d'un ordre total *a priori*
- Écriture : envoi d'un message
- Lecture : exécute toute l'histoire

Exemple



Autres idées envisagées

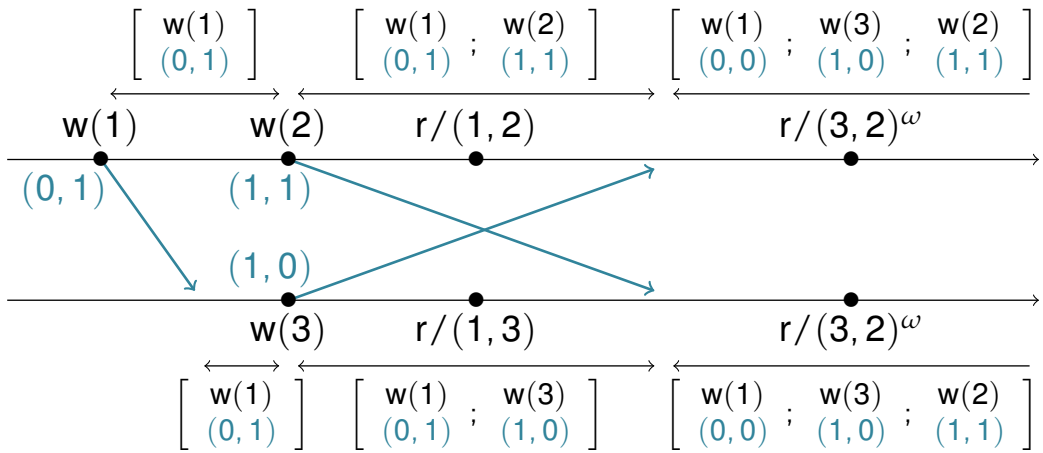
- Exécution locale des opérations à la réception du message
- Convergence asynchrone en envoyant d'autres messages

2. Cohérence d'écritures – Implémentation

Idée

- Construction d'un ordre total *a priori*
- Écriture : envoi d'un message
- Lecture : exécute toute l'histoire

Exemple



Autres idées envisagées

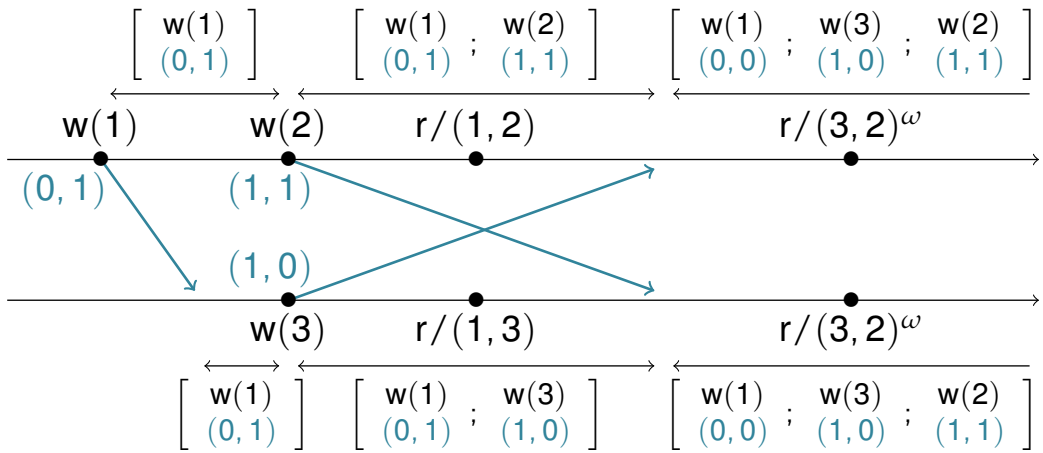
- Exécution locale des opérations à la réception du message
- Convergence asynchrone en envoyant d'autres messages

2. Cohérence d'écritures – Implémentation

Idée

- Construction d'un ordre total *a priori*
- Écriture : envoi d'un message
- Lecture : exécute toute l'histoire

Exemple



Autres idées envisagées

- Exécution locale des opérations à la réception du message
- Convergence asynchrone en envoyant d'autres messages

Critère faible

- Plus faible que la cohérence séquentielle
- Tout objet peut être implémenté sans-attente

Exemples

- Cohérence séquentielle
  - Théorème CAP
- Cohérence d'écritures
- Sérialisabilité
  - Toutes les écritures avortent
- Cohérence pipeline
  - Diffusion FIFO d'un message et exécution locale

Critères primaires

Convergence (EC)

- Tous les processus finissent dans le même état

Validité (V)

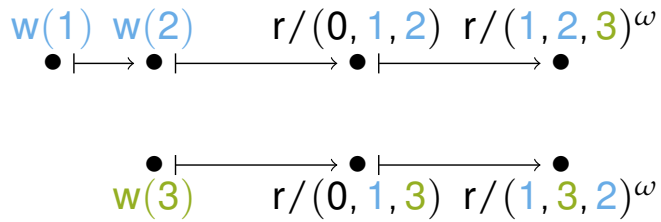
- Chaque processus finit dans un état qui reflète les écritures

Localité d'états (SL)

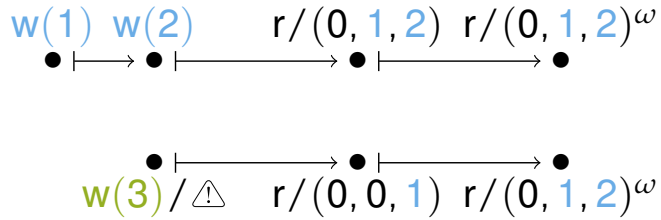
- Tout changement d'état correspond à l'exécution d'une écriture

Critères complémentaires

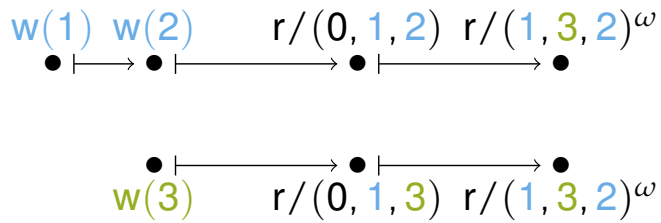
Cohérence pipeline (PC)



Sérialisabilité (Ser)



Cohérence d'écritures (UC)



Critères primaires

Convergence (EC)

- Tous les processus finissent dans le même état

Validité (V)

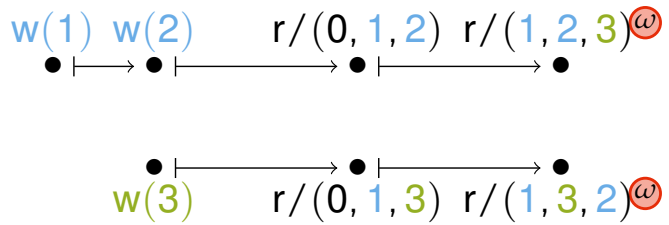
- Chaque processus finit dans un état qui reflète les écritures

Localité d'états (SL)

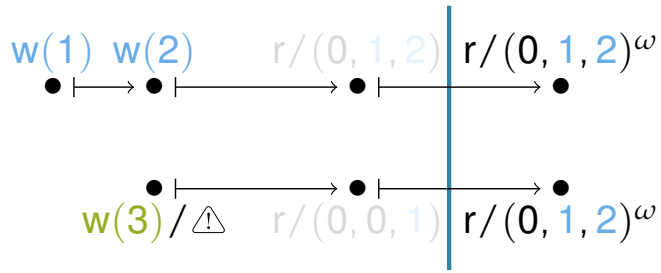
- Tout changement d'état correspond à l'exécution d'une écriture

Critères complémentaires

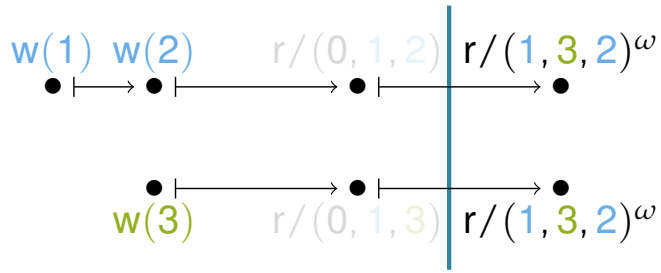
Cohérence pipeline (PC)



Sérialisabilité (Ser)



Cohérence d'écritures (UC)



Critères primaires

Convergence (EC)

- Tous les processus finissent dans le même état

Validité (V)

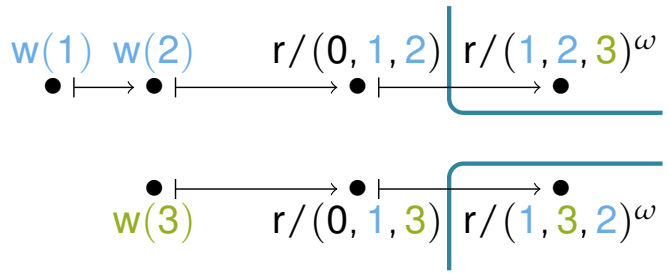
- Chaque processus finit dans un état qui reflète les écritures

Localité d'états (SL)

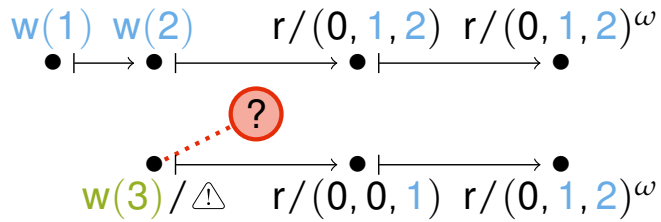
- Tout changement d'état correspond à l'exécution d'une écriture

Critères complémentaires

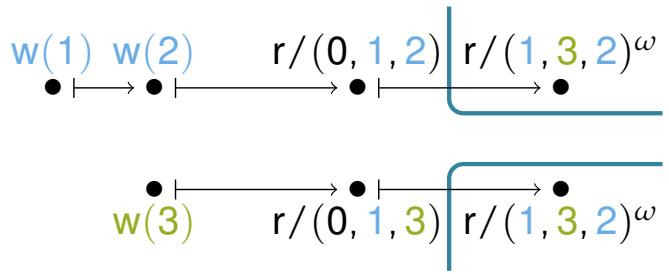
Cohérence pipeline (PC)



Sérialisabilité (Ser)



Cohérence d'écritures (UC)



Critères primaires

Convergence (EC)

- Tous les processus finissent dans le même état

Validité (V)

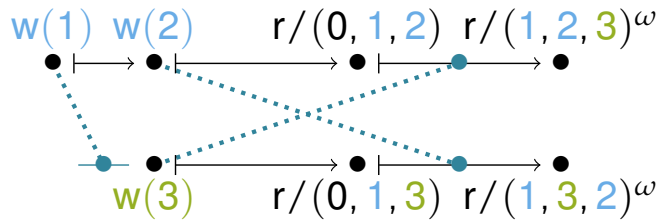
- Chaque processus finit dans un état qui reflète les écritures

Localité d'états (SL)

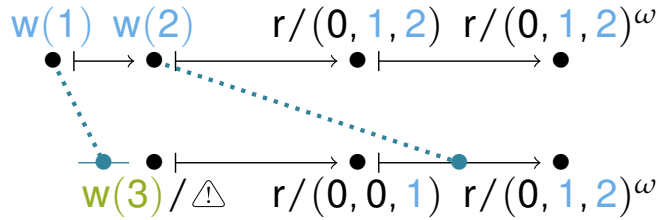
- Tout changement d'état correspond à l'exécution d'une écriture

Critères complémentaires

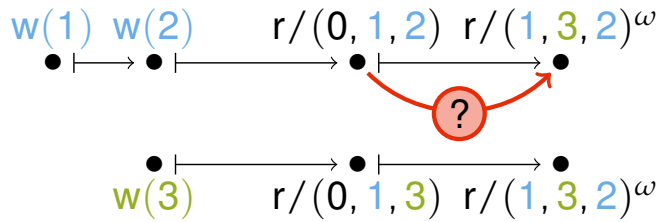
Cohérence pipeline (PC)



Sérialisabilité (Ser)



Cohérence d'écritures (UC)



3. Calculabilité – $EC + V + SL$

Proposition

- $EC + V + SL$ est un critère fort

Preuve

- Flux fenêtré ($EC + V + SL$)-cohérent
- Consensus

Consensus

Terminaison : tout processus correct retourne une valeur
- Localité d'état

Accord : toutes les valeurs retournées sont identiques
- Convergence

Validité : toute valeur retournée a été proposée
- Validité

Impossible d'implémenter le Consensus sans-attente[1]

[1] Fischer, Lynch, Paterson. *Impossibility of distributed consensus with one faulty process*. JACM 1985

3. Calculabilité – $EC + V + SL$

Proposition

- $EC + V + SL$ est un critère fort

Preuve

- Flux fenêtré ($EC + V + SL$)-cohérent
- Consensus

Consensus

Terminaison : tout processus correct retourne une valeur
- Localité d'état

Accord : toutes les valeurs retournées sont identiques
- Convergence

Validité : toute valeur retournée a été proposée
- Validité

Impossible d'implémenter le Consensus sans-attente[1]

[1] Fischer, Lynch, Paterson. *Impossibility of distributed consensus with one faulty process*. JACM 1985

3. Calculabilité – $EC + V + SL$

Proposition

- $EC + V + SL$ est un critère fort

Preuve

- Flux fenêtré ($EC + V + SL$)-cohérent
- Consensus

Consensus

Terminaison : tout processus correct retourne une valeur
- Localité d'état

Accord : toutes les valeurs retournées sont identiques
- Convergence

Validité : toute valeur retournée a été proposée
- Validité

Impossible d'implémenter le Consensus sans-attente[1]

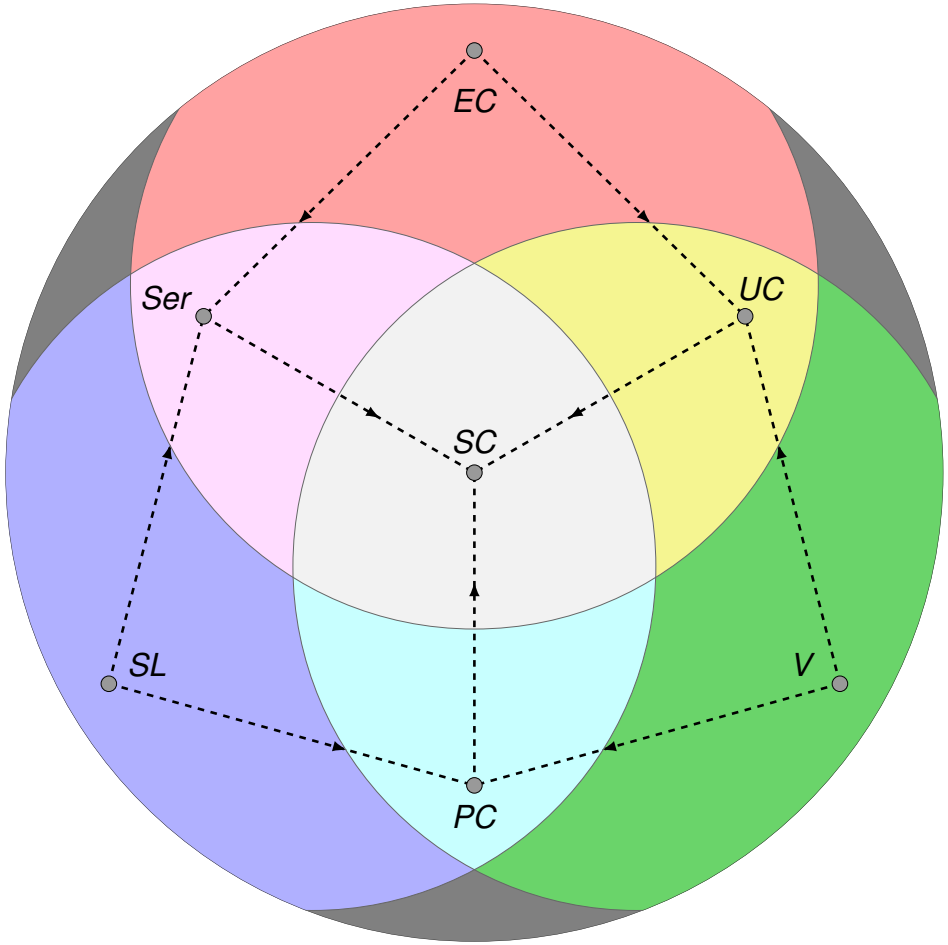
[1] Fischer, Lynch, Paterson. *Impossibility of distributed consensus with one faulty process*. JACM 1985

Critères primaires

- Localité d'état
- Convergence
- Validité

Critères secondaires

- Cohérence d'écritures
- Cohérence pipeline
- Sérialisabilité



Conjonction de critères complémentaires

- Critères forts

Sérialisabilité → substitut à la cohérence forte

- Sécurité maximale
- Implémentation facile dans le modèle client-serveur
- Gestion des pannes par l'utilisateur

Cohérence d'écritures → applications collaboratives, données

- Proche de l'autostabilisation
- Coût relativement élevé
- Incohérences visibles par un utilisateur

Cohérence pipeline → algorithmes parallèles

- Prévisibilité
- Faible coût
- Absence de convergence

3. Calculabilité – Quel critère utiliser ?

Sérialisabilité → substitut à la cohérence forte

- Sécurité maximale
- Implémentation facile dans le modèle client-serveur
- Gestion des pannes par l'utilisateur

Cohérence d'écritures → applications collaboratives, données

- Proche de l'autostabilisation
- Coût relativement élevé
- Incohérences visibles par un utilisateur

Cohérence pipeline → algorithmes parallèles

- Prévisibilité
- Faible coût
- Absence de convergence

3. Calculabilité – *Quel critère utiliser ?*

Sérialisabilité → substitut à la cohérence forte

- Sécurité maximale
- Implémentation facile dans le modèle client-serveur
- Gestion des pannes par l'utilisateur

Cohérence d'écritures → applications collaboratives, données

- Proche de l'autostabilisation
- Coût relativement élevé
- Incohérences visibles par un utilisateur

Cohérence pipeline → algorithmes parallèles

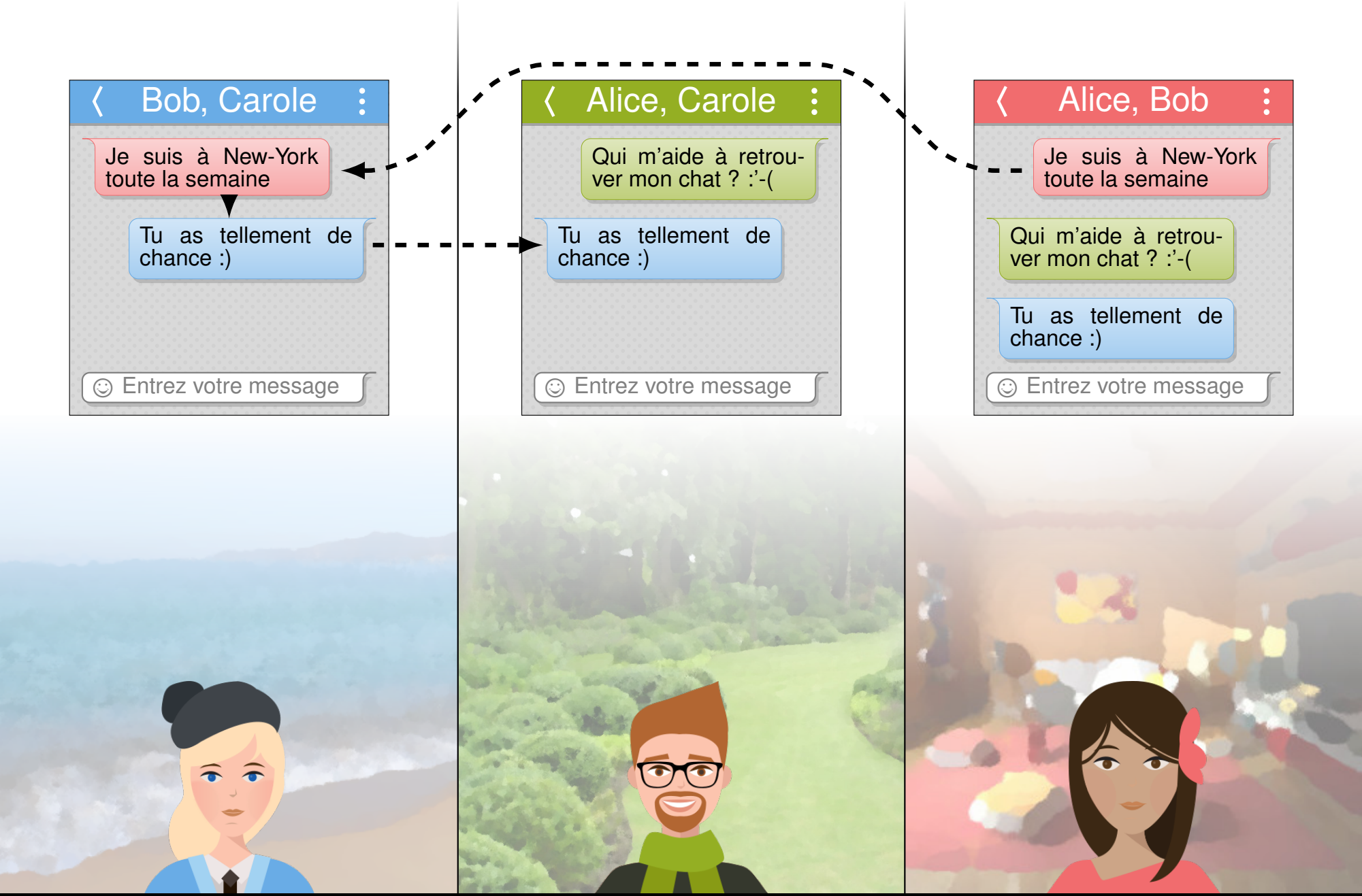
- Prévisibilité
- Faible coût
- Absence de convergence

4. Causalité – Motivation

4. Causalité – Motivation


Bob, Carole
Je suis à New-York toute la semaine
Tu as tellement de chance :)
Entrez votre message
Alice, Carole
Qui m'aide à retrouver mon chat ? :'-(
Tu as tellement de chance :)
Entrez votre message
Alice, Bob
Je suis à New-York toute la semaine
Qui m'aide à retrouver mon chat ? :'-(
Tu as tellement de chance :)
Entrez votre message
Tu as tellement de chance :)

4. Causalité – Motivation



Relation d'écriture-lecture

- Relation $\mathrm{w}_x(n) \multimap \mathrm{r}_x/n$
- Au plus un antécédent par lecture
- Les lectures sans antécédent retournent 0

$\rightarrow$ dépendances sémantiques

Mémoire causale

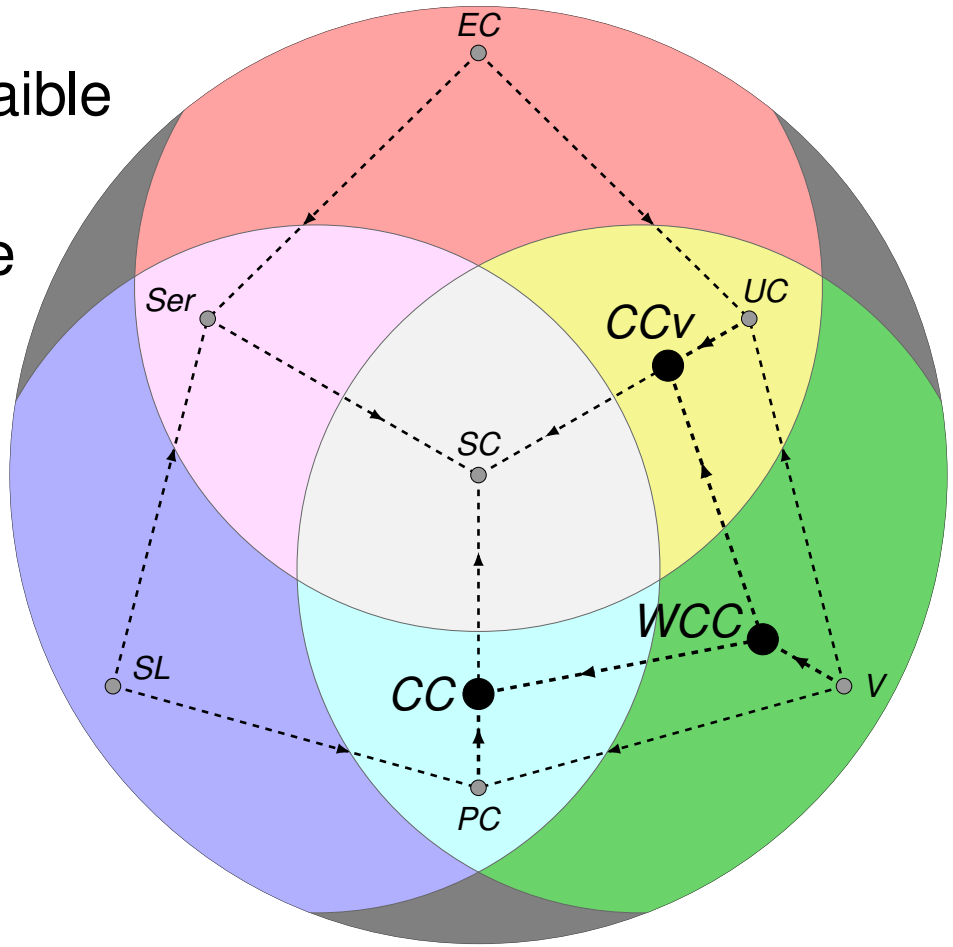
Il existe $\multimap$ tel que

- il existe un ordre partiel $\rightarrow$ qui contient $\multimap$ et $\mapsto$
- Les linéarisations de PRAM respectent $\rightarrow$

[1] Ahamad *et. al. Causal Memory: Definition, Implementation and Programming.* DISC 1995

Trois nouveaux critères

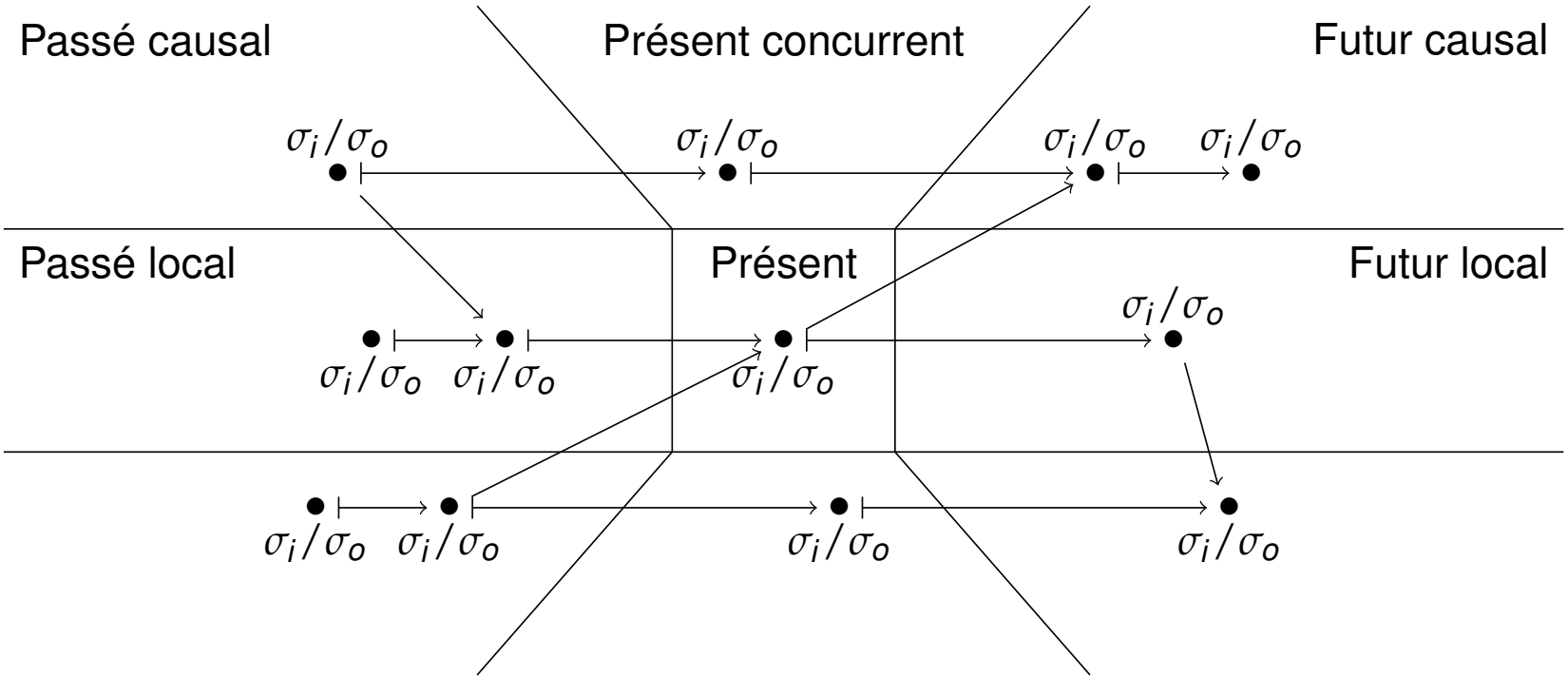
- Cohérence causale
  - $CC \geq PC$
- Cohérence causale faible
  - $WCC \geq V$
- Convergence causale
  - $CCv \geq UC$



Ordre causal

- Ordre partiel sur les événements
- Contient l'ordre des processus

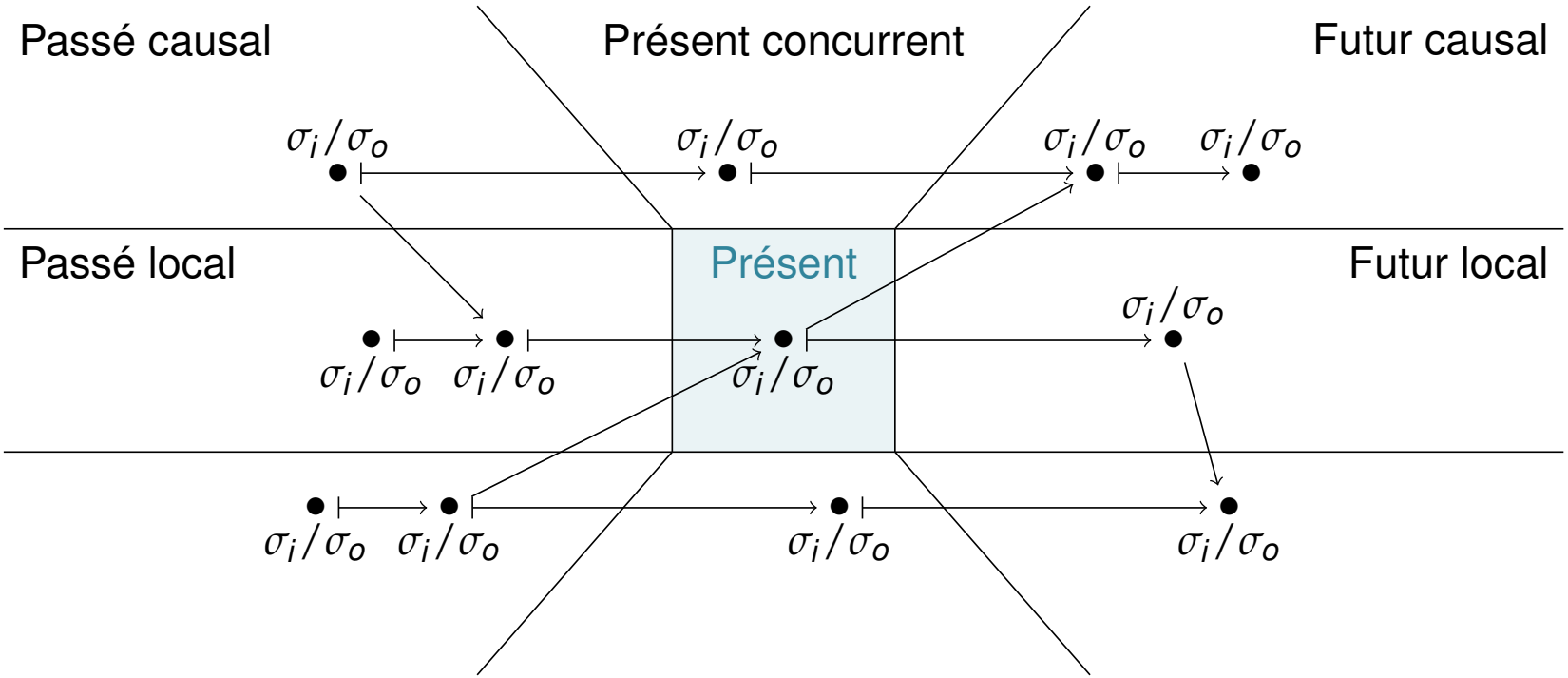
6 zones temporelles par événement



Ordre causal

- Ordre partiel sur les événements
- Contient l'ordre des processus

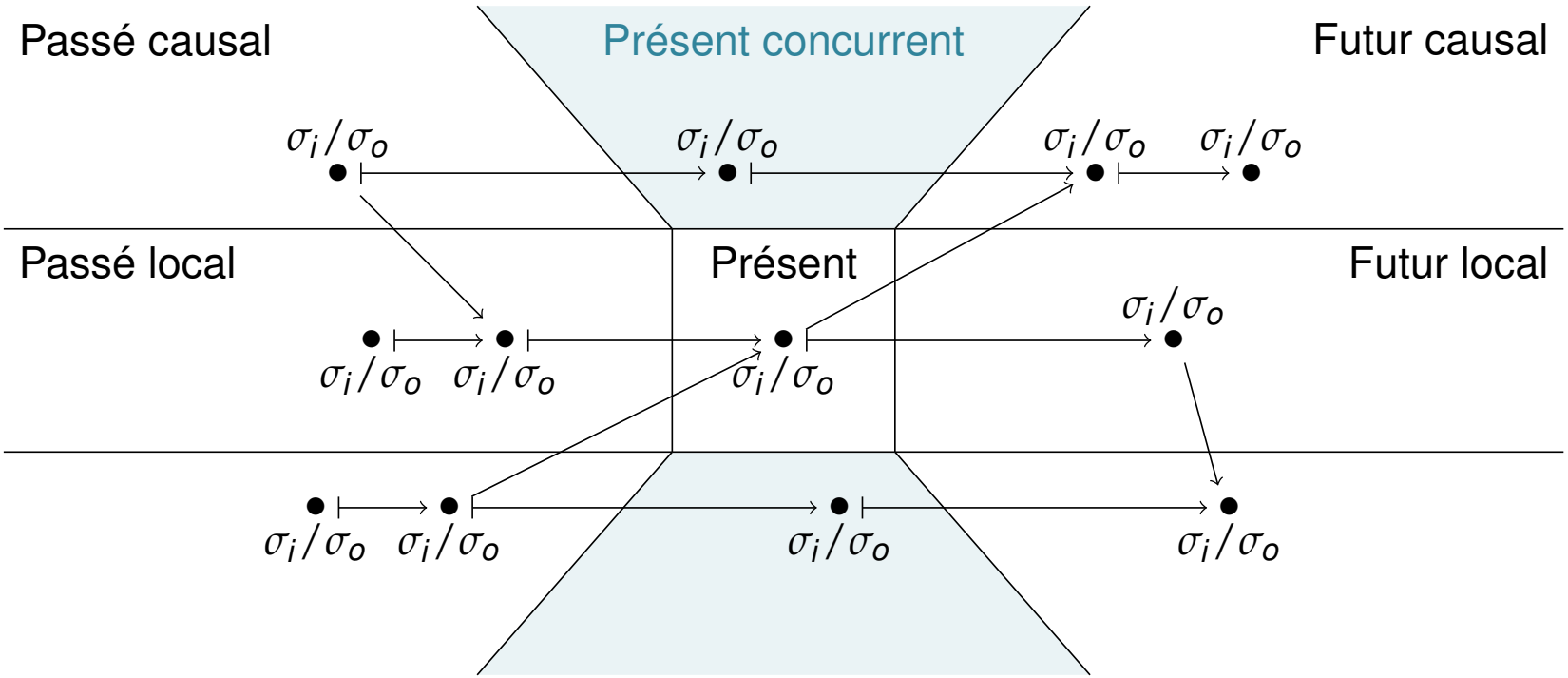
6 zones temporelles par événement



Ordre causal

- Ordre partiel sur les événements
- Contient l'ordre des processus

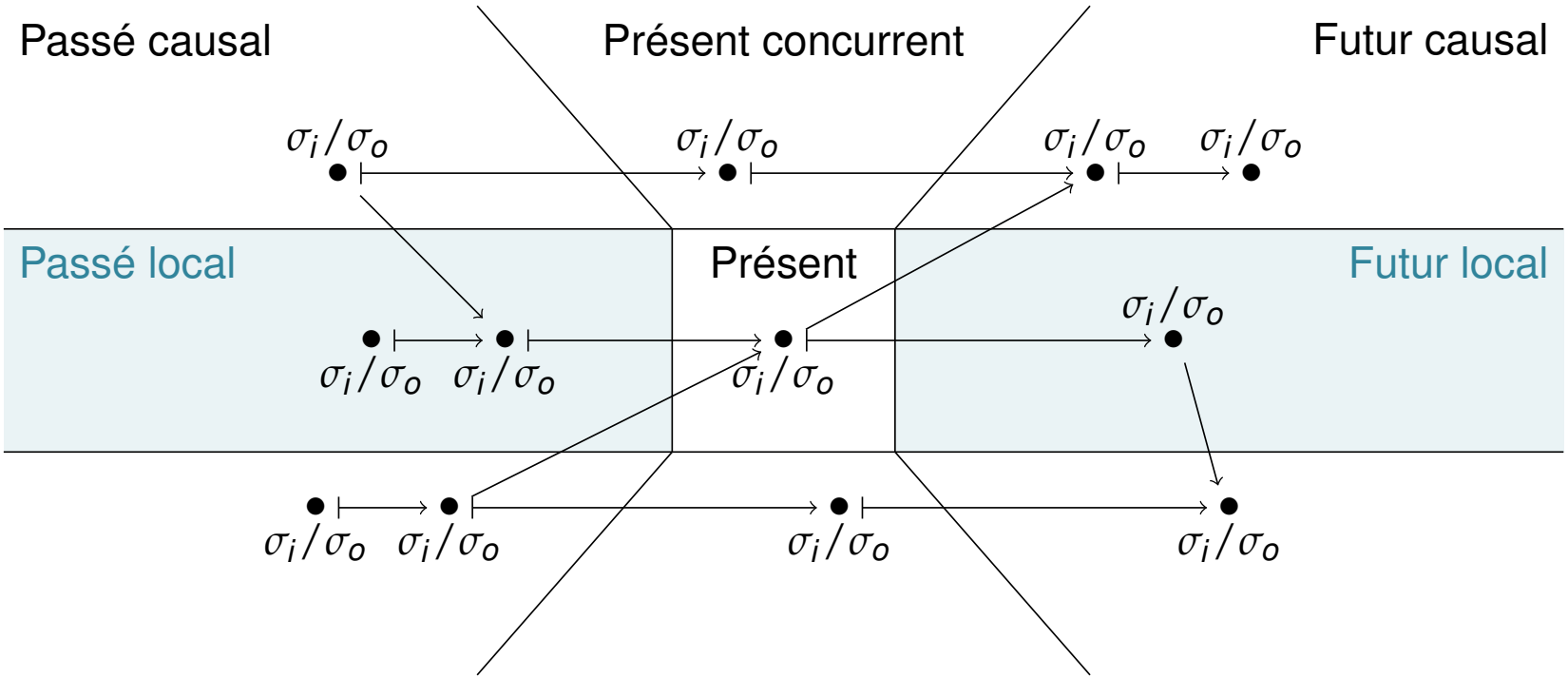
6 zones temporelles par événement



Ordre causal

- Ordre partiel sur les événements
- Contient l'ordre des processus

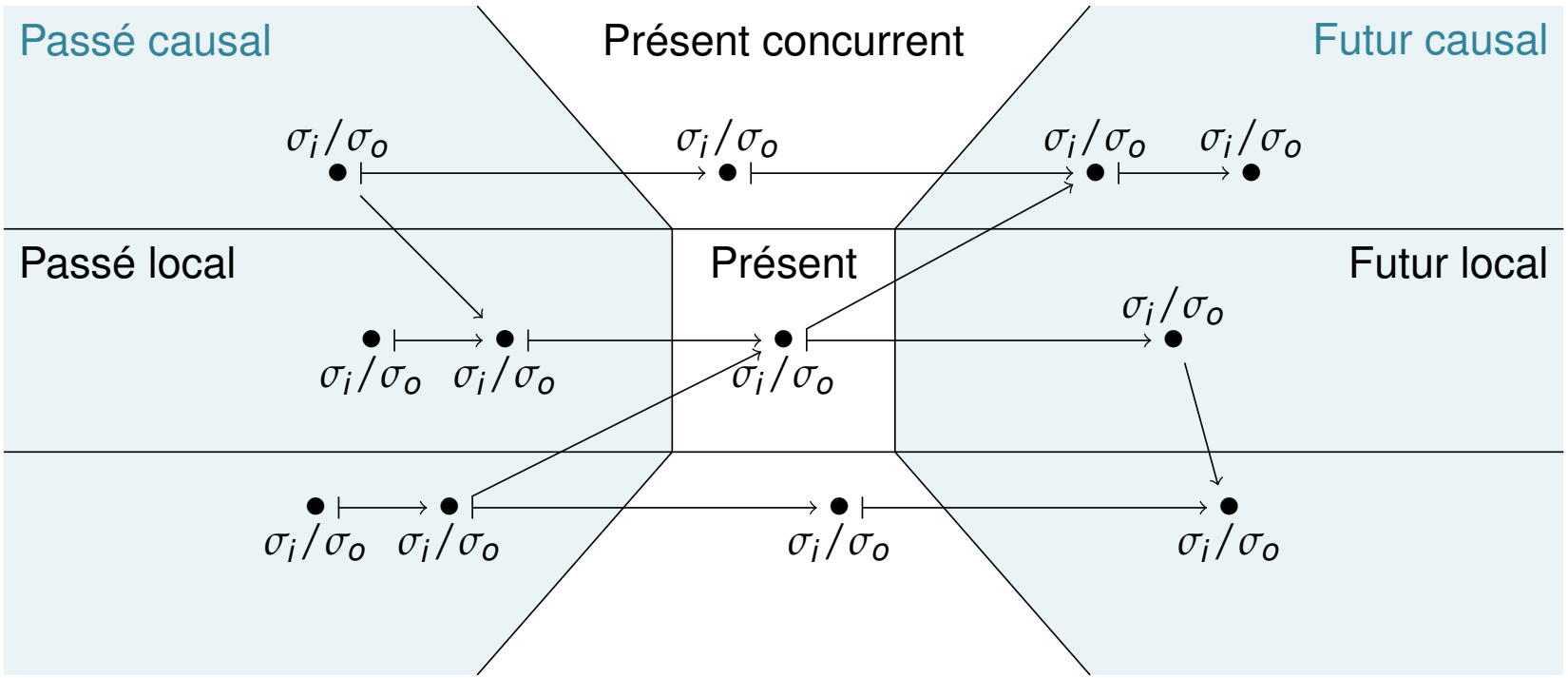
6 zones temporelles par événement



Ordre causal

- Ordre partiel sur les événements
- Contient l'ordre des processus

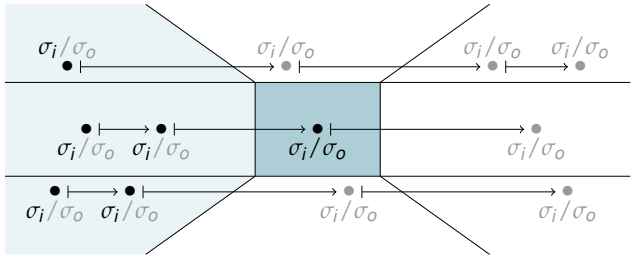
6 zones temporelles par événement



Cohérence causale faible

Il existe :

- Un ordre causal
- Une linéarisation par événement
  - qui respecte l'ordre causal
  - qui contient le présent
  - qui contient les écritures du passé causal
  - admise par la spécification séquentielle



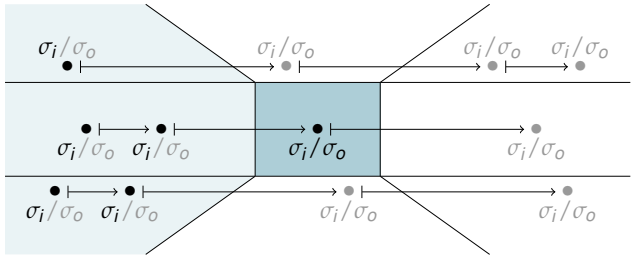
Cohérence causale

- Les linéarisations contiennent le passé local

Cohérence causale faible

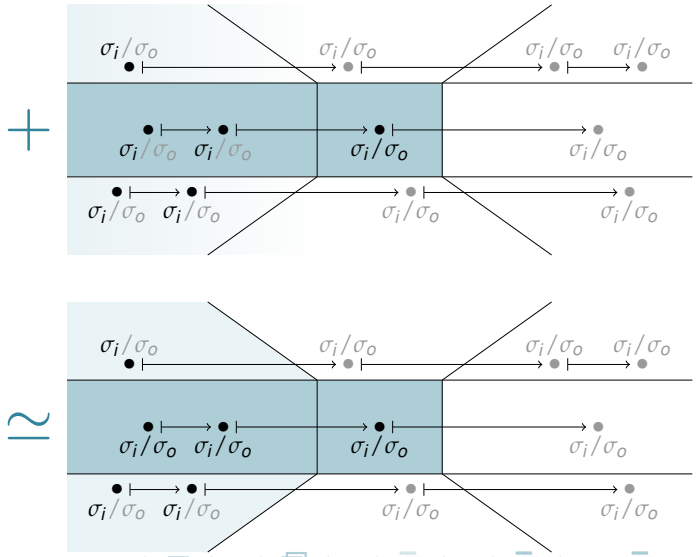
Il existe :

- Un ordre causal
- Une linéarisation par événement
  - qui respecte l'ordre causal
  - qui contient le présent
  - qui contient les écritures du passé causal
  - admise par la spécification séquentielle



Cohérence causale

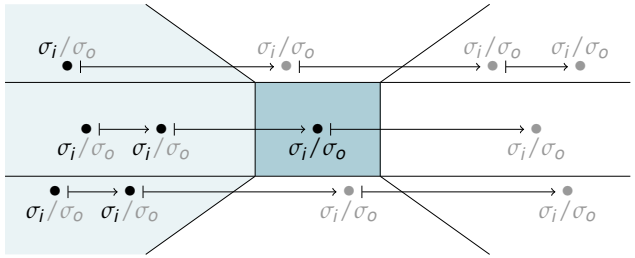
- Les linéarisations contiennent le passé local



Cohérence causale faible

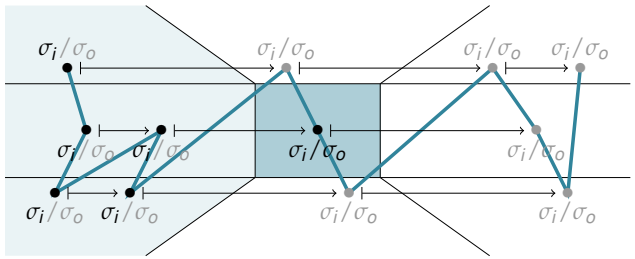
Il existe :

- Un ordre causal
- Une linéarisation par événement
  - qui respecte l'ordre causal
  - qui contient le présent
  - qui contient les écritures du passé causal
  - admise par la spécification séquentielle

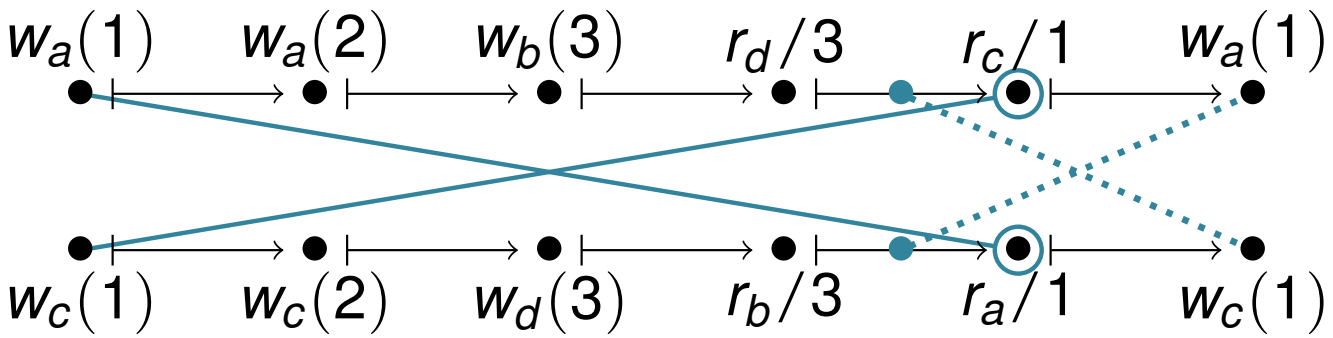


Convergence causale

- Les linéarisations respectent un ordre total commun



Mémoire causale ✓ Cohérence causale X



Proposition

- H causalement cohérente pour la mémoire
- H admise par la mémoire causale

Réciproque

- Toutes les écritures de H sont différentes [1]
- H admise par la mémoire causale
- H causalement cohérente pour la mémoire

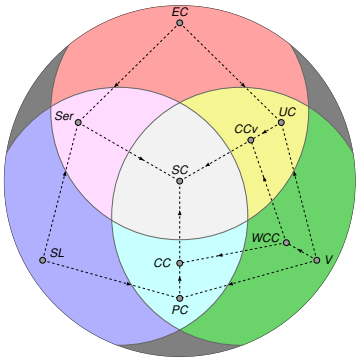
[1] Misra. *Axioms for Memory Access in Asynchronous Hardware Systems.* TOPLAS 1986

Conclusion – Proposition

Spécification séquentielle


```
class Messenger {
   string[] received;
   void send(string message) {
     received.length ++ ;
     received[received.length - 1] = message ;
   }
}
```

Critère de cohérence




```
void main () {
   Messenger m = UC.connect!Messenger("AliceBob");
         ⋮
}
```

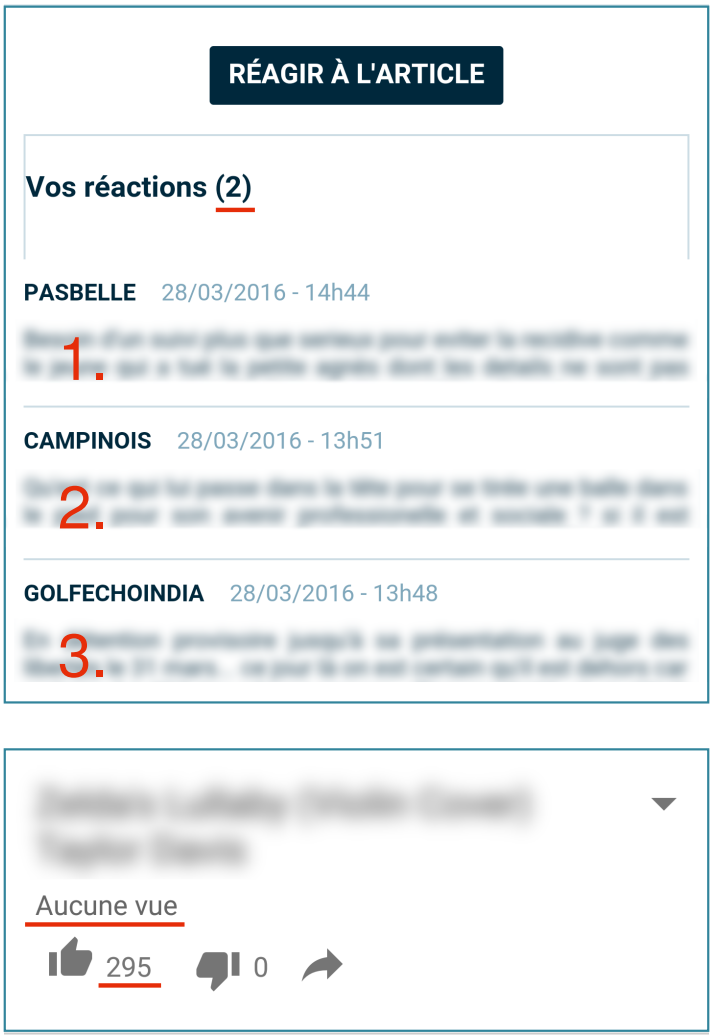
Calculabilité

- Identification d'un objet impossible à implémenter
- Implémentations génériques

Perspectives

- Implémentations adaptées aux objets
  - Étude de la complexité
- Génération de code optimal
  - Dans le cadre de CODS
- Se rapprocher de la cohérence forte en pratique
  - Mesure quantitative de la cohérence

Conclusion – Programmes composés de plusieurs objets



Composabilité

- Propriété très rare

Perspectives

- Composition d'objets
- Composition de critères
- Contraintes d'intégrité[1]
  - Lien avec les bases de données

[1] Maussion. *Update Consistency for Data Integrity in Distributed Data Bases*. Rapport de stage, 2015

Conclusion – Publications

PPoPP'16 : P. M. J. *Causal Consistency: Beyond Memory*. 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2016

IPDPS'15 : P. M. J. *Update Consistency for Wait-free Concurrent Objects*. 29th IEEE International Parallel and Distributed Processing Symposium, 2015

NETYS'15 : P. J. M. *Tracking Causal Dependencies in Web Services Orchestrations Defined in ORC*. 3rd International Conference on NETwork sYStems, 2015

DISC'14 : P. M. J. *Brief Announcement: Update Consistency in Partitionable Systems*. 28th International Symposium on Distributed Computing, 2014

MSR'13 : P. J. M. *Construction d'une sémantique concurrente par instrumentation d'une sémantique opérationelle structurelle*. Modélisation des Systèmes Réactifs, 2013

IJFCS : B. P. T. *On the Complexity of Concurrent Multiset Rewriting*. International Journal of Foundations of Computer Sciences, 2015