darcs, utilisation et questions ouvertes

Florent Becker

LIFO & Darcs Team

Séminaire LIFO

Plan

1. Comment on s'en sert
2. Comment ca marche ?
3. Où ça va ?
   - eXtra Meilleurs Lendemains
   - La paix dans le monde

Qu'est-ce que c'est ?

- `http://darcs.net`

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)
  - Pas de serveur central.

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)
  - Pas de serveur central.
  - Fonctionnement symétrique

Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)
  - Pas de serveur central.
  - Fonctionnement symétrique
- Simple

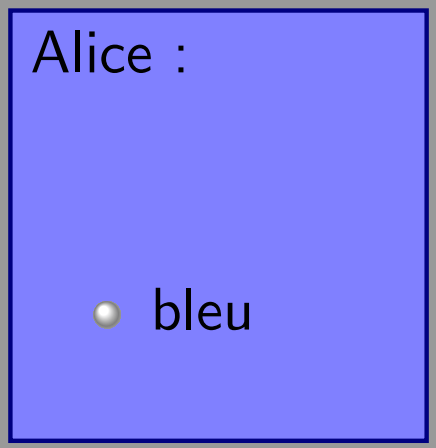
Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)
  - Pas de serveur central.
  - Fonctionnement symétrique
- Simple
  - Utilisation interactive

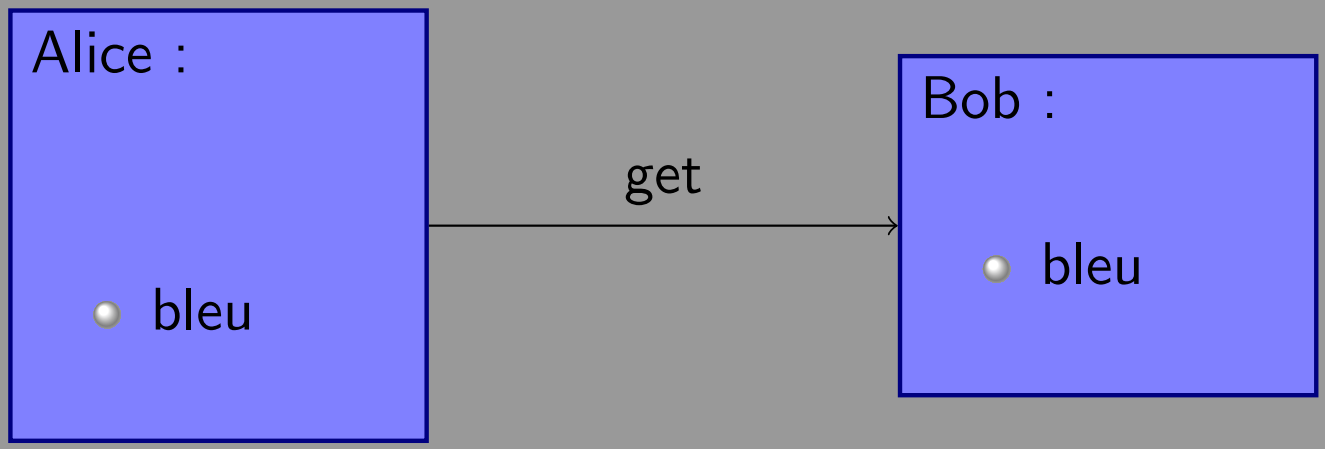
Qu'est-ce que c'est ?

- `http://darcs.net`
- Gestion de Versions :
  - Alice et Bob travaillent sur un fichier ensemble, comment synchroniser leur travail ?
  - Quand Zbigniew arrive, le système tient-il ?
  - Mises à jour concurrentes
- Distribuée (en pair à pair)
  - Pas de serveur central.
  - Fonctionnement symétrique
- Simple
  - Utilisation interactive
  - Fonctionnement souple

Un exemple



Un exemple



Un exemple

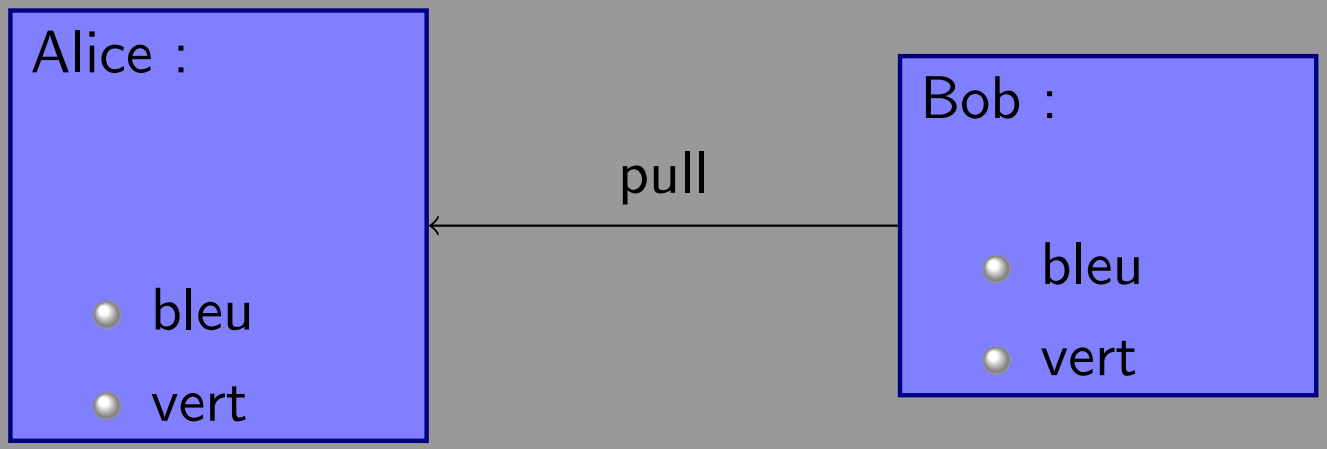
Alice :

- bleu

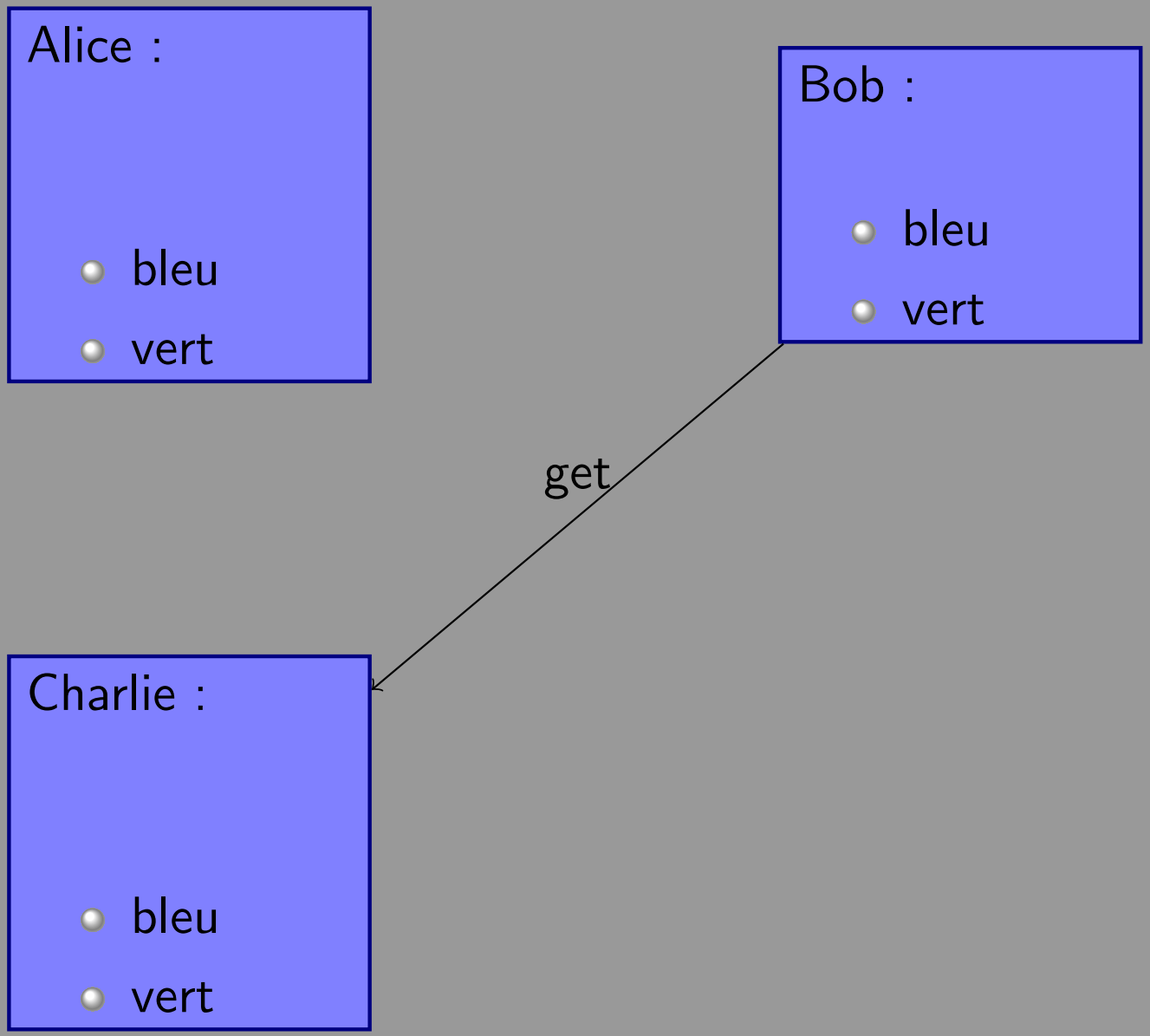
Bob :

- bleu
- vert

Un exemple



Un exemple



Un exemple

Alice :

- bleu
- vert

Bob :

- bleu
- vert

Charlie :

- bleu
- vert

Un exemple

Alice :

- bleu
- vert

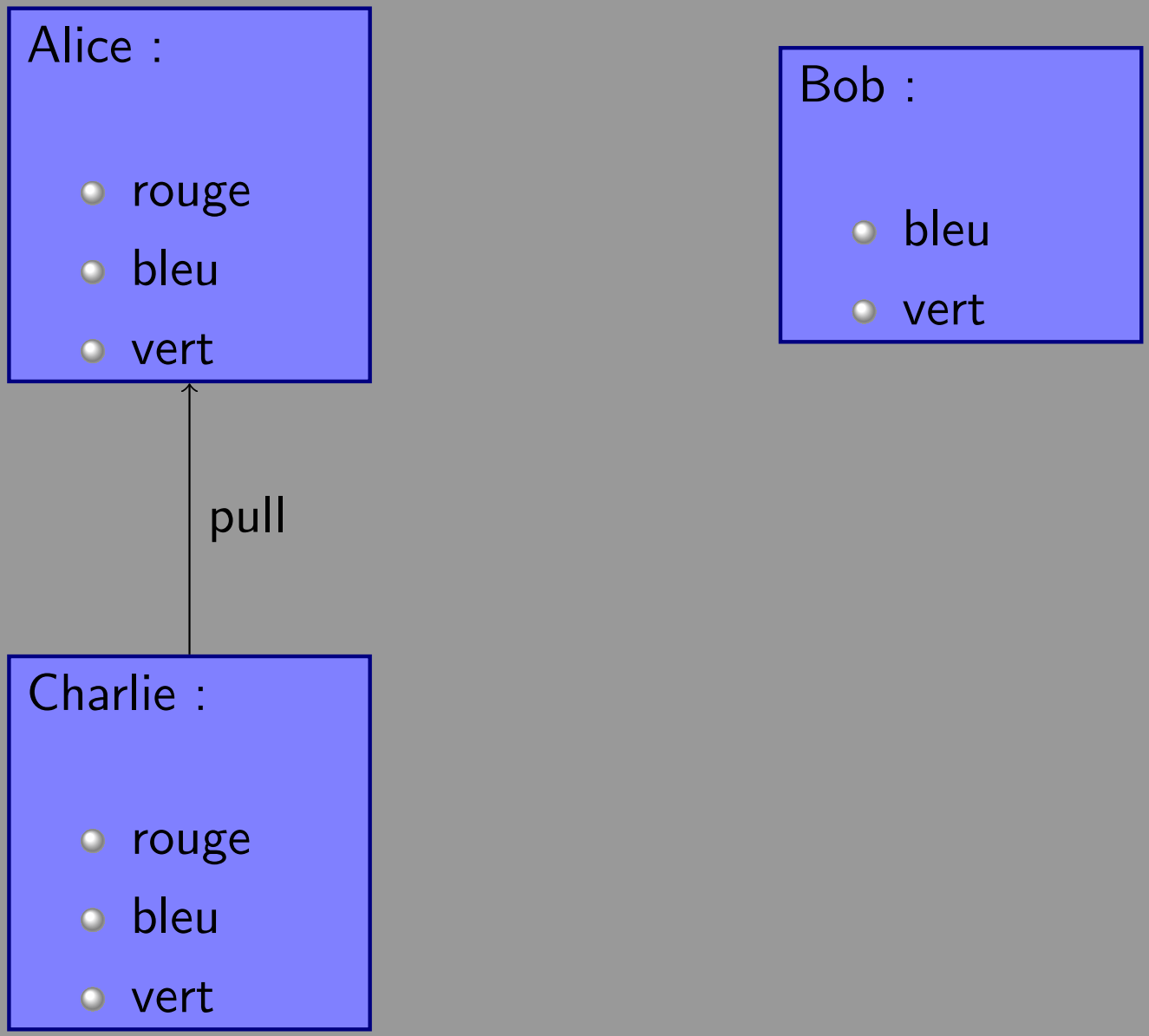
Bob :

- bleu
- vert

Charlie :

- rouge
- bleu
- vert

Un exemple


Alice :
rouge
bleu
vert
Bob :
bleu
vert
pull
Charlie :
rouge
bleu
vert

Un exemple

Alice :

- rouge
- bleu
- vert

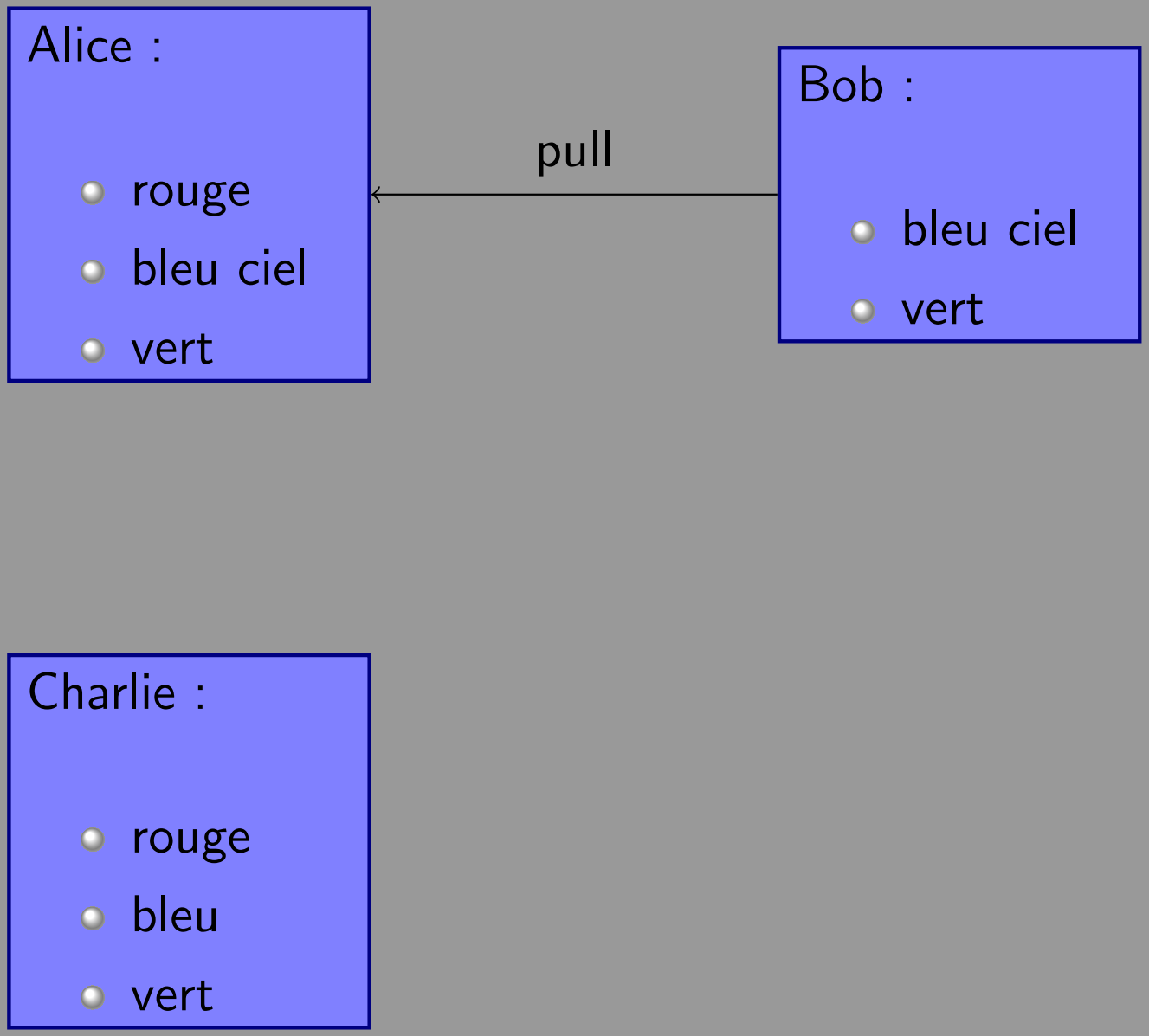
Bob :

- bleu ciel
- vert

Charlie :

- rouge
- bleu
- vert

Un exemple


Alice :
rouge
bleu ciel
vert
pull
Bob :
bleu ciel
vert
Charlie :
rouge
bleu
vert

Dans la console

- init
- get
- record
- pull
- send/push

Différentes organisations

- Serveur central, comme cvs
- Tout par mail
- ≪ Façon linux ≫

1. Comment on s'en sert
2. Comment ca marche ?
3. Où ça va ?
   - eXtra Meilleurs Lendemains
   - La paix dans le monde

Barrière ou piquet ?

- Dans cvs, svn ou git, on garde un historique des versions d'un fichier
- Dans `darcs`, les auteurs s'échangent leurs modifications des fichiers en ≪pair à pair≫, *pas nécessairement dans l'ordre*
- *amendements* plutôt que *code civil*
- Possibilité unique et naturelle de `darcs` : récupérer une *partie* des changements des autres.
- Solution *formelle* au problème du merge (pas de `diff3`).

Changement et contexte

Un changement n'a de sens que dans un certain contexte :

Example

À l'alinéa 10, substituer aux mots :
« se déclarent »,
les mots :
« déposent une liste de ».

Changement et contexte

Un changement n'a de sens que dans un certain contexte :

Example

À l'alinéa 10, substituer aux mots :
« se déclarent »,
les mots :
« déposent une liste de ».

Change de sens après :

Example

Un amendement précédent...
Après l'alinéa 4, insérer un alinéa rédigé comme suit : « [...] »

Changement et Patch

Definition

Un changement est une modification du *sens* du contenu de nos fichiers. C'est une fonction *partielle* sur les contenus, que l'on peut interpréter.

Example

Cet amendement a pour objet de remplacer le vote sur des sigles des organisations syndicales, proposé par le projet de loi, par un vote sur des listes de candidats représentants les organisations syndicales, afin de favoriser l'intérêt des salariés des très petites entreprises pour cette nouvelle élection et le développement du dialogue social.

Changement et Patch

Definition

Un changement est une modification du *sens* du contenu de nos fichiers.
C'est une fonction *partielle* sur les contenus, que l'on peut interpréter.

Definition

Un patch est la représentation de ce changement dans un certain contexte.

Changement et Patch

Definition

Un changement est une modification du *sens* du contenu de nos fichiers.
C'est une fonction *partielle* sur les contenus, que l'on peut interpréter.

Definition

Un patch est la représentation de ce changement dans un certain contexte.

Example

À l'alinéa 10, substituer aux mots : « se déclarent », les mots :
« déposent une liste de ».

Changement et Patch

Definition

Un changement est une modification du *sens* du contenu de nos fichiers. C'est une fonction *partielle* sur les contenus, que l'on peut interpréter.

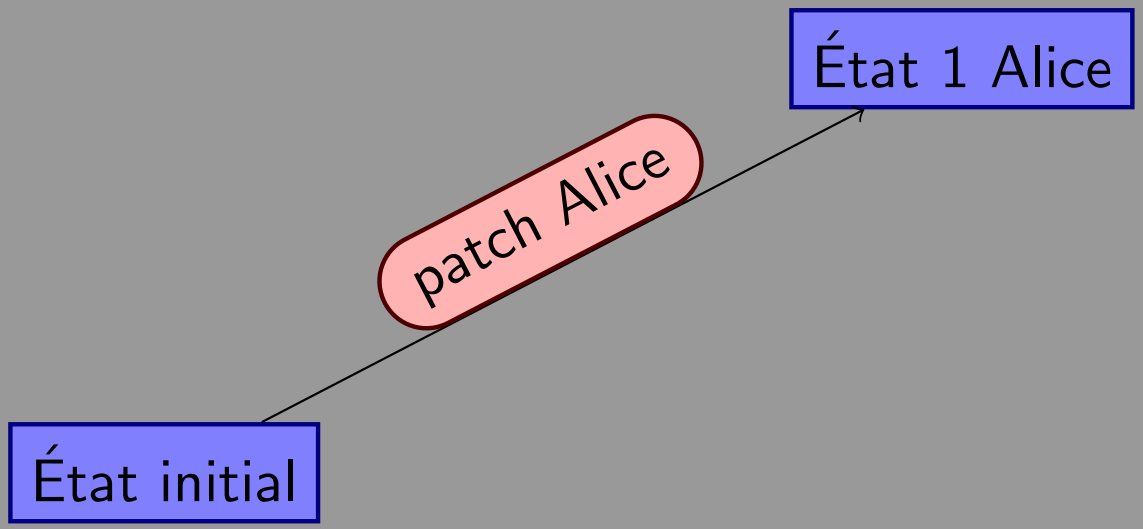
Definition

Un patch est la représentation de ce changement dans un certain contexte.

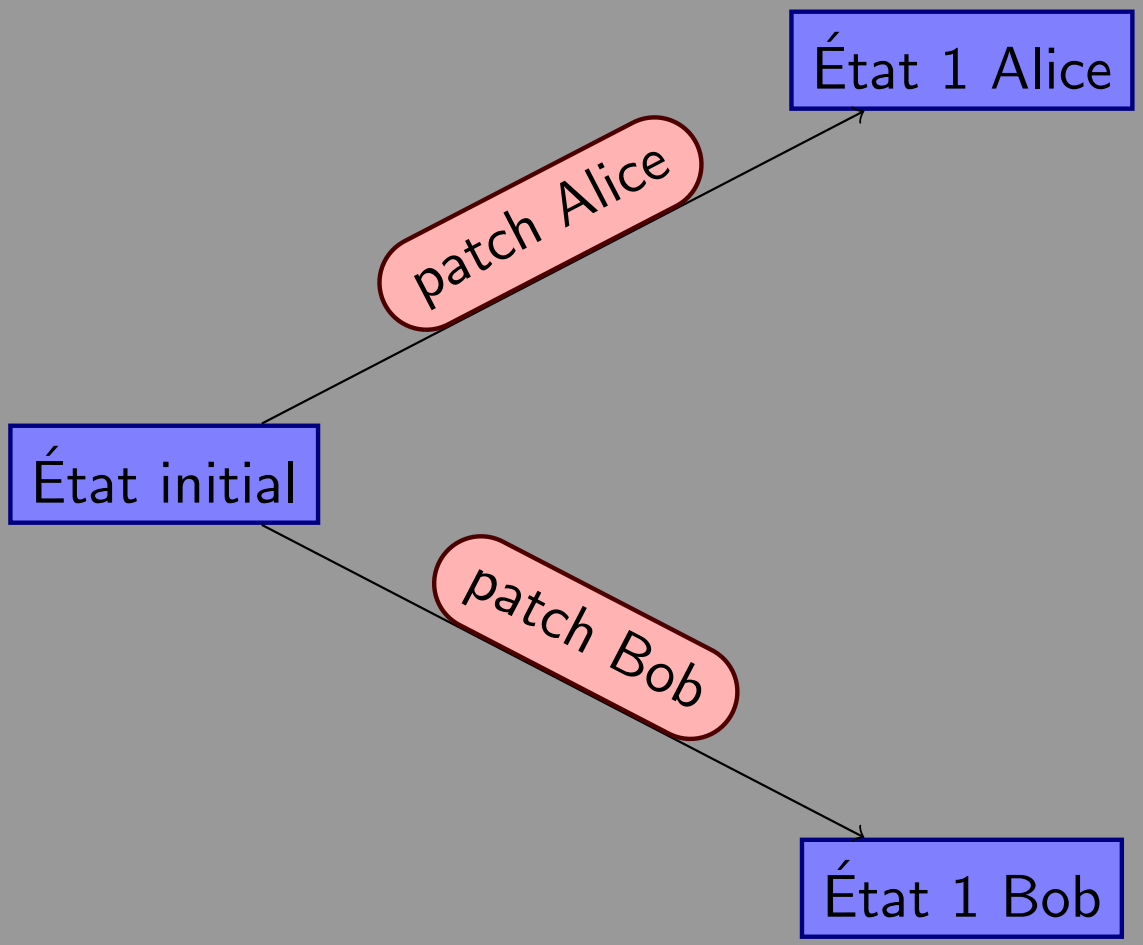
Definition

Un dépôt est une *suite de patches*, qui représente un *ensemble de changements*.

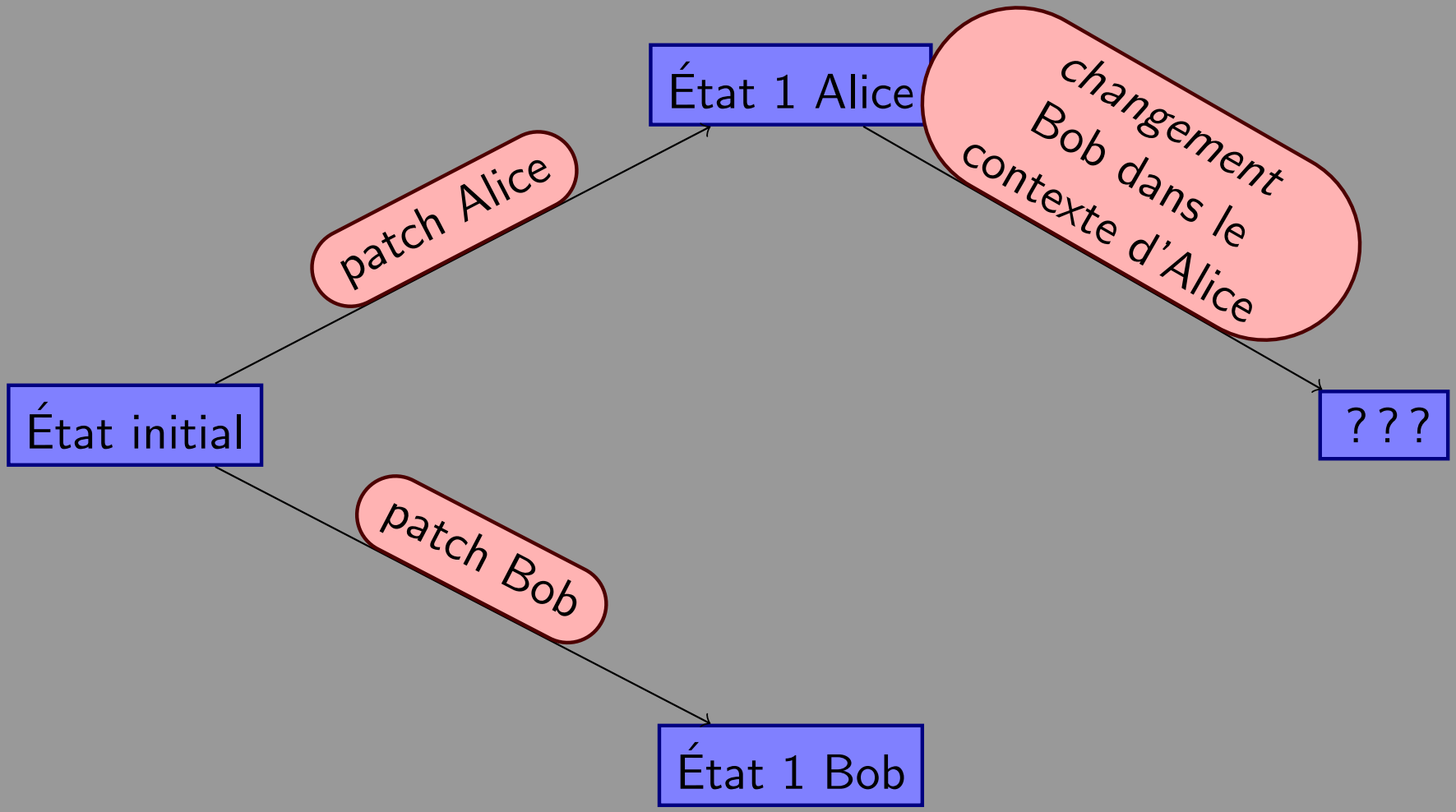
Le problème du merge



Le problème du merge



Le problème du merge

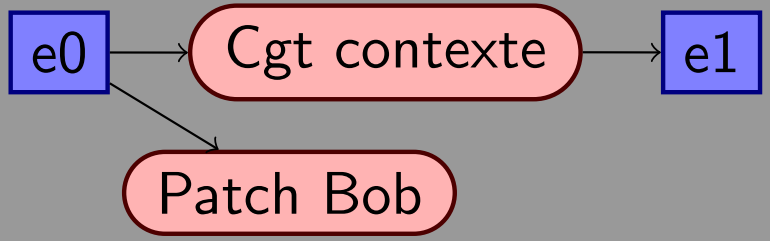


Changements de contexte

- Passer d'un contexte à un autre, c'est appliquer un patch, ici celui d'Alice

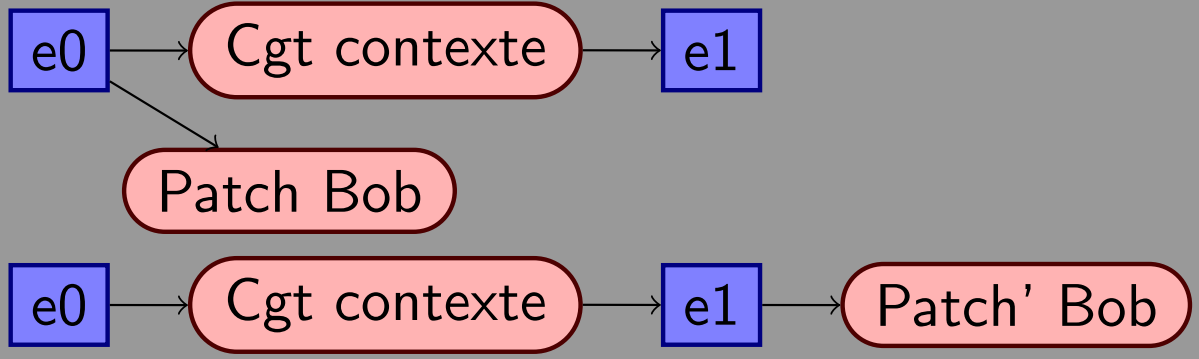
Changements de contexte

- Passer d'un contexte à un autre, c'est appliquer un patch, ici celui d'Alice

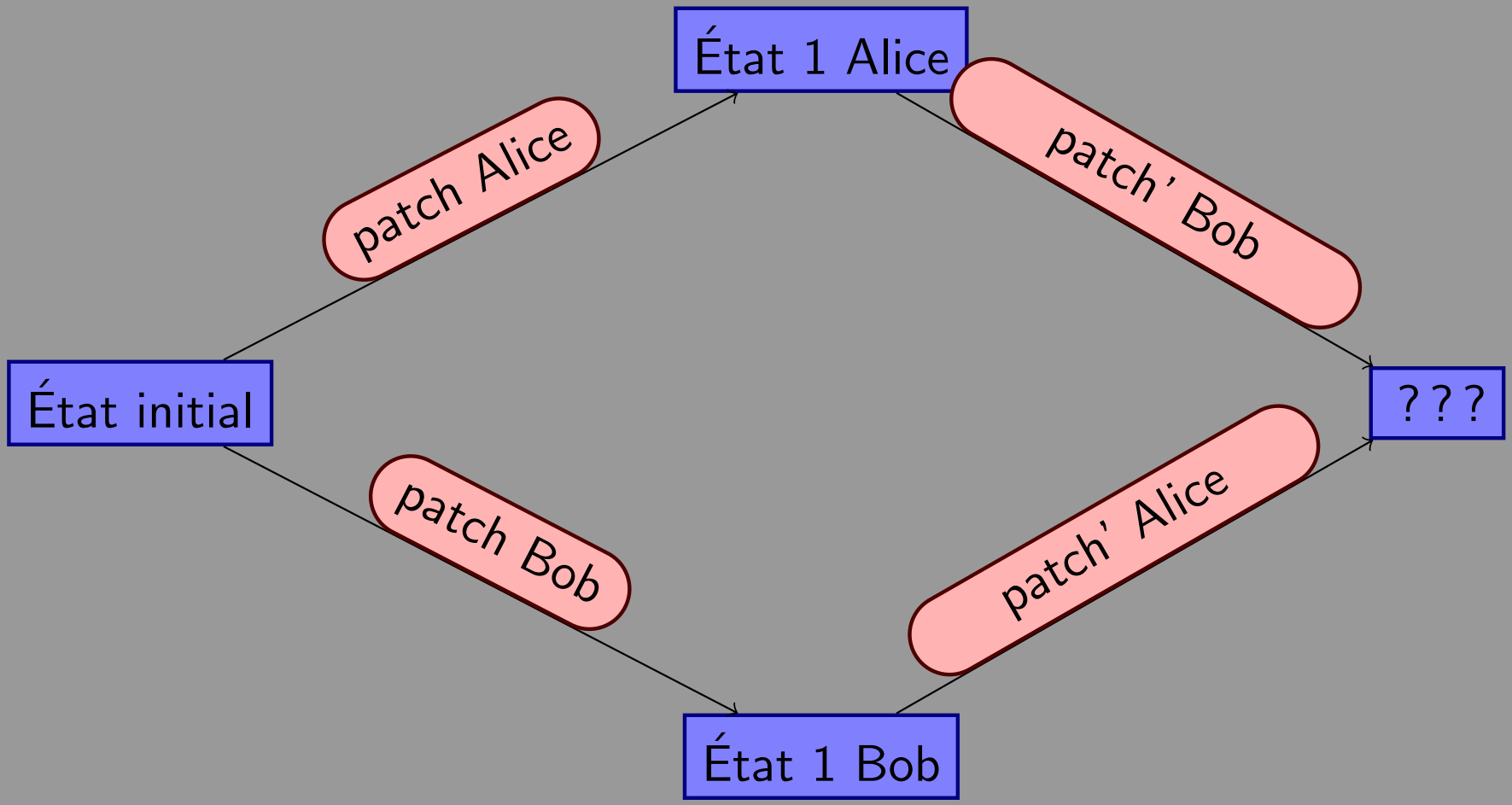


Changements de contexte

- Passer d'un contexte à un autre, c'est appliquer un patch, ici celui d'Alice



Par symétrie :



Commutation

- Pour Alice : patch Alice patch' Bob

Commutation

- Pour Alice : patch Alice, patch' Bob
- Pour Bob : patch Bob, patch' Alice

Commutation

- Pour Alice : patch Alice, patch' Bob
- Pour Bob : patch Bob, patch' Alice
- Les deux sont cohérents : $pA \circlearrowleft pB' = (pB, pA')$

Commutation

- Pour Alice : patch Alice, patch' Bob
- Pour Bob : patch Bob, patch' Alice
- Les deux sont cohérents : $pA \circlearrowleft pB' = (pB, pA')$
- On résout l'équation : $pA^{-1} \circlearrowleft pB = (pB', pA'^{-1})$

Les deux opérations de base

- *Tous* les patches sont inversibles
- On a une opération de commutation *partielle*

Non-commutation

Example

- Patch 0 : `ajouter un fichier toto.txt`
- Patch 1 : `ajouter au début de toto.txt: ‘‘vert’’`

- 0 et 1 ne commutent pas, car on ne peut pas modifier un fichier qui n'existe pas

Non-commutation

Example

- Patch 0 : `ajouter un fichier toto.txt`
- Patch 1 : `ajouter au début de toto.txt: ‘‘vert’’`
- Patch 2 : `ligne 1 de toto.txt: ‘‘vert’’ -> ‘‘rouge’’`

- 0 et 1 ne commutent pas, car on ne peut pas modifier un fichier qui n'existe pas
- 1 et 2 ne commutent pas, car on ne peut pas modifier une ligne qui n'existe pas (inversibilité)

Non-commutation

Example

- Patch 0 : `ajouter un fichier toto.txt`
- Patch 1 : `ajouter au début de toto.txt: ‘‘vert’’`
- Patch 2 : `ligne 1 de toto.txt: ‘‘vert’’ -> ‘‘rouge’’`
- Patch 3 : `ligne 1 de toto.txt: ‘‘rouge’’ -> ‘‘bleu’’`

- 0 et 1 ne commutent pas, car on ne peut pas modifier un fichier qui n'existe pas
- 1 et 2 ne commutent pas, car on ne peut pas modifier une ligne qui n'existe pas (inversibilité)
- 2 et 3 ne commutent pas : on s'intéresse au changement en entier, pas à son état final

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
  - Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
  - Dépôt de Bérénice : `p1 p2 p3 p4 pB1' pB2 pB3`

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
  - Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
  - Dépôt de Bérénice : `p1 p2 p3 p4 pB1' pB2 pB3`
- Aurélien veut pB1, pB3 mais pas pB2.
  - dépôt de Bérénice : `p1 p2 p3 p4 pB1' pB3' pB2'`
  - Commutation impossible = dépendances

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
- Aurélien veut pB1, pB3 mais pas pB2.
  - dépôt de Bérénice : `p1 p2 p3 p4 pB1' pB3' pB2'`
  - Commutation impossible = dépendances
- On intègre chez Aurélien
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2` $pA2^{-1}$ $pA1^{-1}$ `pB1' pB3'`

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
- Aurélien veut pB1, pB3 mais pas pB2.
- On intègre chez Aurélien
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2` $pA2^{-1}$ $pA1^{-1}$ `pB1' pB3'`
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2 pB1'' pB3''` $pA2'^{-1}$ $pA1'^{-1}$

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
- Aurélien veut pB1, pB3 mais pas pB2.
- On intègre chez Aurélien
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2` $pA2^{-1}$ $pA1^{-1}$ `pB1' pB3'`
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2 pB1'' pB3''` $pA2'^{-1}$ $pA1'^{-1}$
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2 pB1'' pB3''`

darcs pull

Que se passe-t-il quand Aurélien tape : `darcs pull http://berenice/depot` ?

- Dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2`
- Dépôt de Bérénice : `p1 p3' p2' pB1 p4' pB2 pB3`
- Alignement des dépots.
- Aurélien veut pB1, pB3 mais pas pB2.
- On intègre chez Aurélien
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2` $pA2^{-1}$ $pA1^{-1}$ `pB1' pB3'`
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2 pB1'' pB3''` $pA2'^{-1}$ $pA1'^{-1}$
  - dépôt d'Aurélien : `p1 p2 p3 p4 pA1 pA2 pB1'' pB3''`
  - Commutation impossible = conflit

Conclusion sur la théorie

- Deux opération à définir : inverse et commutation.
- L'algorithme de fusion fonctionne indépendamment du type de fichiers.
- Commutation partielle : patches modulaires. (`darcs replace`)

1. Comment on s'en sert
2. Comment ca marche ?
3. Où ça va ?
   - eXtra Meilleurs Lendemains
   - La paix dans le monde

Perspectives d'utilisation

- Travail collaboratif sur documents structurés (wiki++)
- IHM pour l'édition de documents (undo en mieux)
- Combinateurs abstraits pour les transformations en XML (édition automatique)

Patches en XML

- Définir des opérations sur les arbres
- Niveau 0 : préserver la correction de l'arbre

- Garder la ≪mise en page≫ hors de la structure d'arbre ?

Patches en XML

- Définir des opérations sur les arbres
- Niveau 0 : préserver la correction de l'arbre
- Niveau 1 : opérations définies en termes d'arbre

- Garder la ≪mise en page≫ hors de la structure d'arbre ?

Patches en XML

- Définir des opérations sur les arbres
- Niveau 0 : préserver la correction de l'arbre
- Niveau 1 : opérations définies en termes d'arbre
- Niveau 2 : opérations en fonction de la DTD
- Garder la ≪mise en page≫ hors de la structure d'arbre ?

Que faire face à un conflit

- Représentation interne de la situation
- Informer l'utilisateur
- Représenter la résolution

commute partiel, merge total

- Pour fusionner les patches pA et pB, il faut que $pA^{-1} \circlearrowleft pB$ soit défini.
- On veut que `darcs pull` soit une opération totale.
- Il faut donc compléter $\circlearrowleft$

commute total

On peut chercher à définir un système de patches sur lesquels $\circlearrowleft$ soit totale. Pour cela, on travaille sur des pseudo-fichiers.

- Un système avec inodes et noms de fichiers
- Des lignes à coordonnées rationelles
- Ligne = fonction de $\mathbb{Z}$ dans $\texttt{ASCII}^*$

Problèmes : interface utilisateur, dépendances, comment faire autre chose que du ligne à ligne ?

Patches et conflicteurs

L'autre solution (utilisée dans `darcs`) c'est d'enrichir l'ensemble existant des patches pour rendre $\circlearrowleft$ totale. On appelle *conflicteurs* les patches ainsi ajoutés.

- On conserve la sémantique des patches existants
- Il est facile de repérer les conflits
- Conflicteur naïf : on y garde tout le dépôt
- Conflicteurs imbriqués : quand les résolutions sont en conflit

Problème : complexité de $\circlearrowleft$ avec des conflicteurs.

- Dans darcs 1 : $O(2^n)$, n nombre de conflits

Patches et conflicteurs

L'autre solution (utilisée dans `darcs`) c'est d'enrichir l'ensemble existant des patches pour rendre $\circlearrowleft$ totale. On appelle *conflicteurs* les patches ainsi ajoutés.

- On conserve la sémantique des patches existants
- Il est facile de repérer les conflits
- Conflicteur naïf : on y garde tout le dépôt
- Conflicteurs imbriqués : quand les résolutions sont en conflit

Problème : complexité de $\circlearrowleft$ avec des conflicteurs.

- Dans darcs 1 : $O(2^n)$, n nombre de conflits
- Dans darcs 2 : $O(2^d)$, profondeur du pire conflit (utilisable)

Conclusion

- Un système décentralisé

Conclusion

- Un système décentralisé
- simple

Conclusion

- Un système décentralisé
- simple
- avec des possibilités uniques

Conclusion

- Un système décentralisé
- simple
- avec des possibilités uniques
- des questions intéressantes

Conclusion

- Un système décentralisé
- simple
- avec des possibilités uniques
- des questions intéressantes
- Venez en discuter vendredi (en salle M2)

Conclusion

- Un système décentralisé
- simple
- avec des possibilités uniques
- des questions intéressantes
- Venez en discuter vendredi (en salle M2)
- Merci !