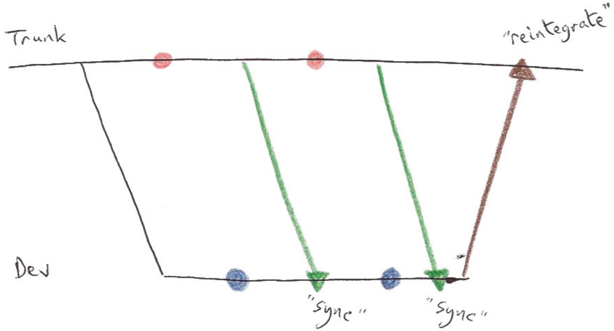
Merge and Performance Improvements

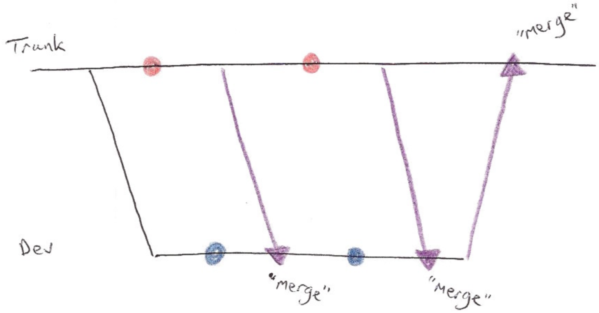
To-and-Fro Merging

WANdisco presents
Subversion Live 2012
San Francisco, CA - Greenwich, CT - London, UK

The “reintegrate” option



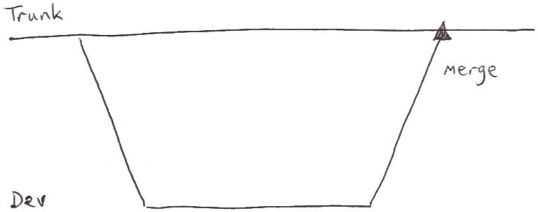
The symmetric merge



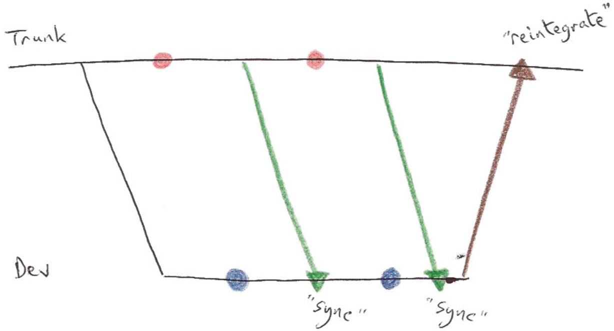
Outline

Outline

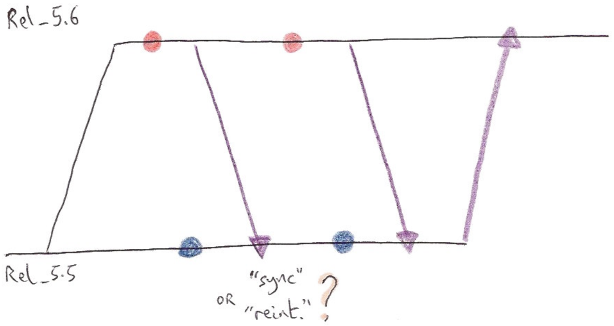
Feature Branch



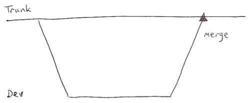
Feature Branch with Sync



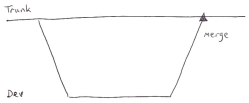
Release Branch

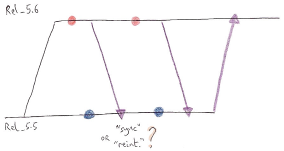


Other Patterns

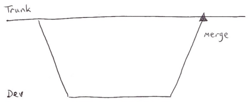


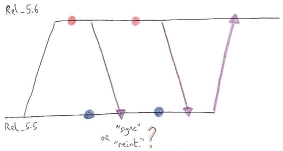
Other Patterns

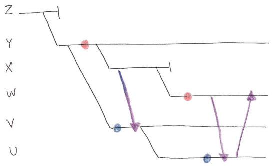




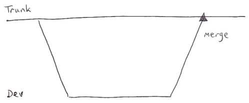
Other Patterns

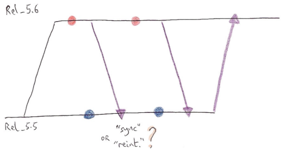


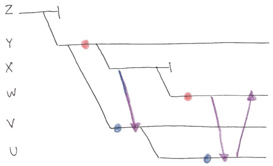


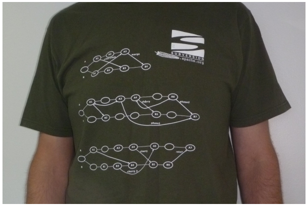


Other Patterns

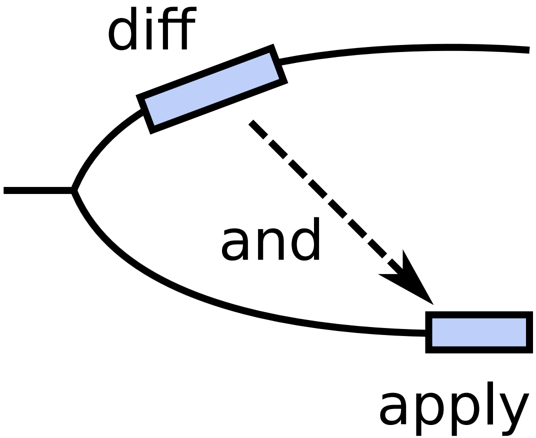




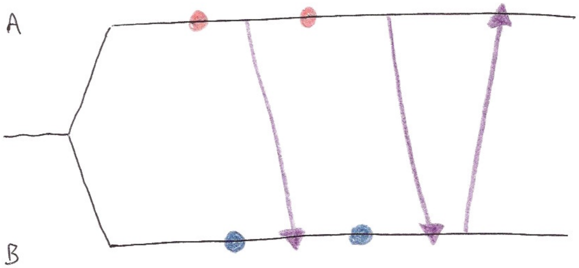




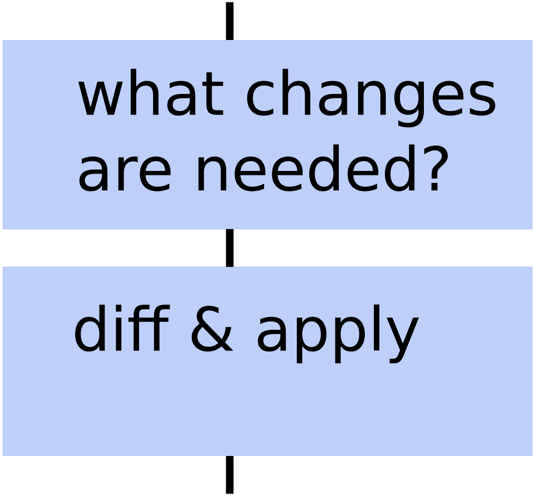
Diff & Apply



Merge Which Changes?

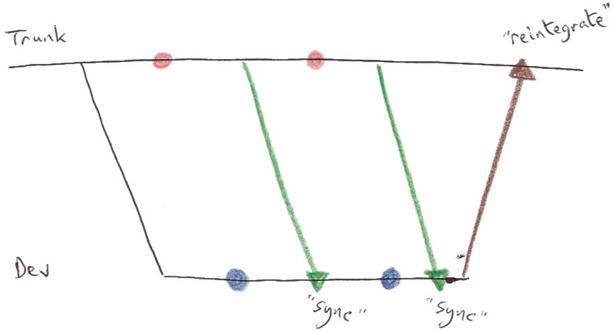


Subsystems

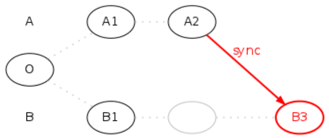


Outline

Sync & Reintegrate

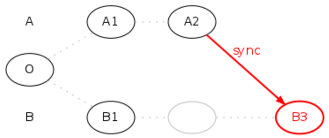


How Sync Works



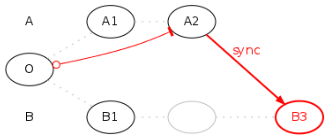
Youngest Common Ancestor	O	O
all changes on source	A1, A2	A1, A2, A3, A4
target's mergeinfo	nil	A:1-2
eligible changes	A1, A2	A3, A4
3-way base	pred(A1)	pred(A3)
3-way source-right	A2	A4
record mergeinfo	"A:1-2"	"A:3-4"

How Sync Works



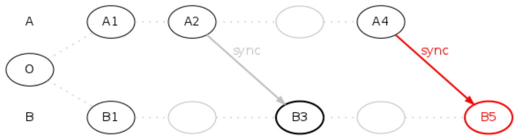
Youngest Common Ancestor	O	O
all changes on source	A1, A2	A1, A2, A3, A4
target's mergeinfo	nil	A:1-2
eligible changes	A1, A2	A3, A4
3-way base	pred(A1)	pred(A3)
3-way source-right	A2	A4
record mergeinfo	"A:1-2"	"A:3-4"

How Sync Works



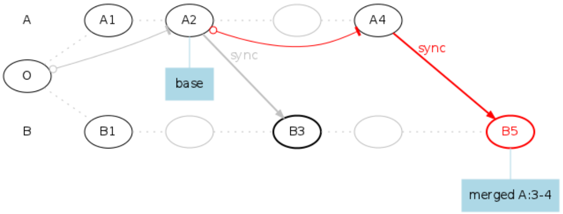
Youngest Common Ancestor	O	O
all changes on source	A1, A2	A1, A2, A3, A4
target's mergeinfo	nil	A:1-2
eligible changes	A1, A2	A3, A4
3-way base	pred(A1)	pred(A3)
3-way source-right	A2	A4
record mergeinfo	"A:1-2"	"A:3-4"

How Sync Works



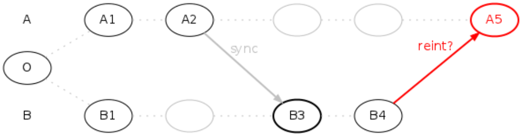
Youngest Common Ancestor	O	O
all changes on source	A1, A2	A1, A2, A3, A4
target's mergeinfo	nil	A:1-2
eligible changes	A1, A2	A3, A4
3-way base	pred(A1)	pred(A3)
3-way source-right	A2	A4
record mergeinfo	"A:1-2"	"A:3-4"

How Sync Works



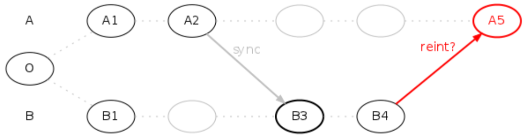
Youngest Common Ancestor	O	O
all changes on source	A1, A2	A1, A2, A3, A4
target's mergeinfo	nil	A:1-2
eligible changes	A1, A2	A3, A4
3-way base	pred(A1)	pred(A3)
3-way source-right	A2	A4
record mergeinfo	"A:1-2"	"A:3-4"

How Reintegrate Works



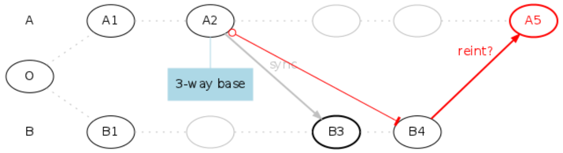
Youngest Common Ancestor .	O
all changes on *target* .	A1, A2, A3, A4
source's mergeinfo .	A:1-2
eligible changes .	diff(A2, B4)
3-way base .	A2
3-way source-right .	B4
record mergeinfo .	"B:1-4"

How Reintegrate Works



Youngest Common Ancestor	.	O
all changes on *target*	.	A1, A2, A3, A4
source's mergeinfo	.	A:1-2
eligible changes	.	diff(A2, B4)
3-way base	.	A2
3-way source-right	.	B4
record mergeinfo	.	"B:1-4"

How Reintegrate Works



Youngest Common Ancestor	O
all changes on *target*	A1, A2, A3, A4
source's mergeinfo	A:1-2
eligible changes	diff(A2, B4)
3-way base	A2
3-way source-right	B4
record mergeinfo	"B:1-4"

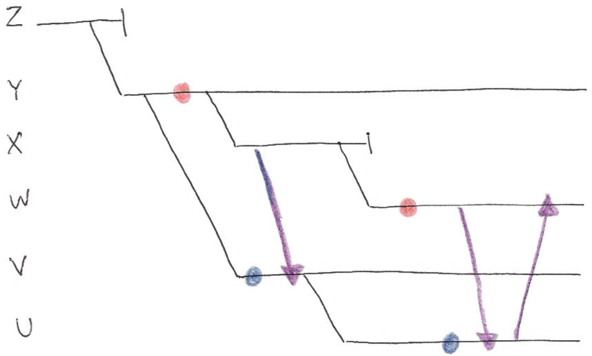
Differences

Table: Differences

	sync	reintegrate
base node	on *source* branch	on *target* branch
skip cherry-picked revs?	yes	no
fill in partly-merged subtrees?	yes	no
handle local mods in the WC?	yes	no

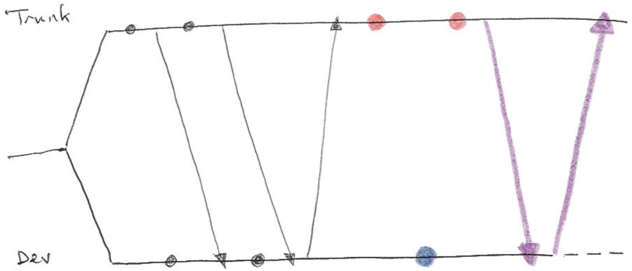
Outline

Confusing



Outline

Continue



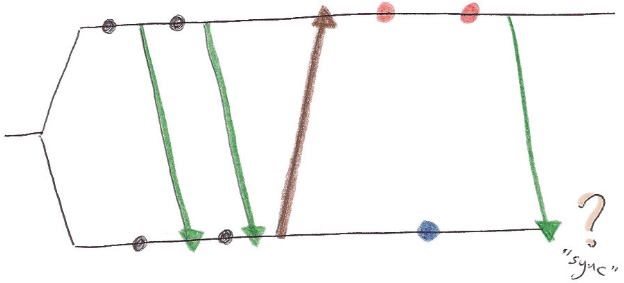
Continue

- Delete
- Keep Alive

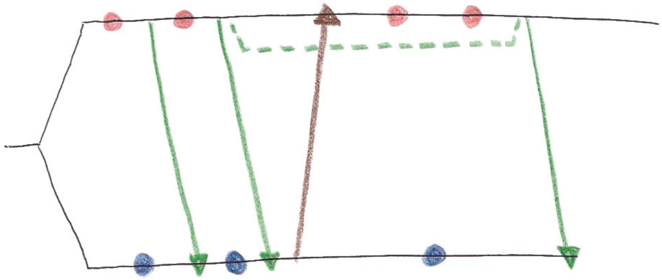
Continue

- Delete
- Keep Alive

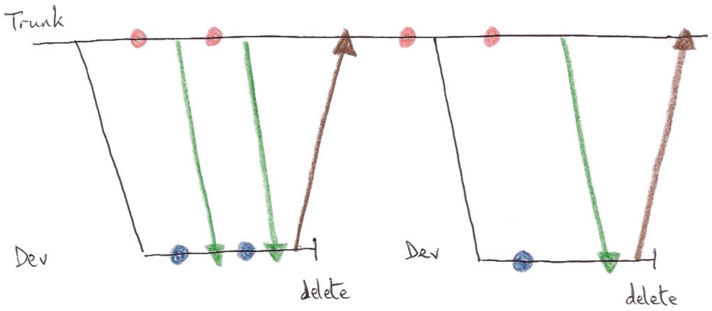
Continue



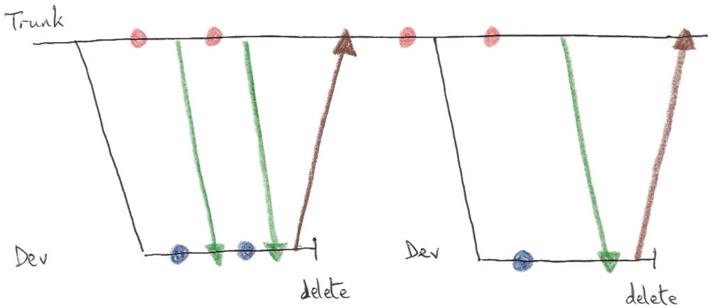
Continue



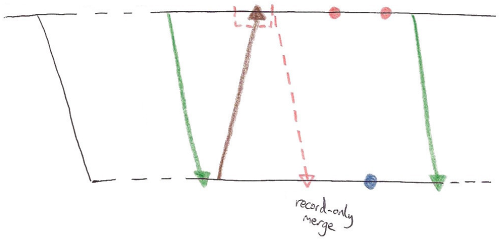
Delete & re-branch



Delete & re-branch

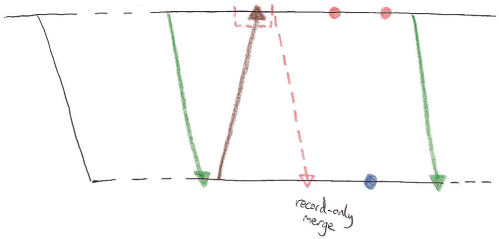


Keep Alive



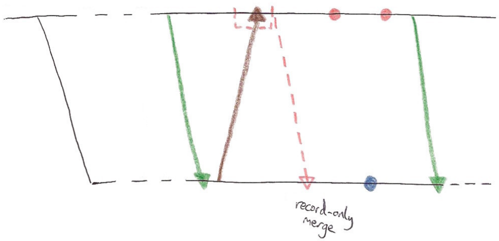
- Awkward extra step
- Doesn't work properly, in general

Keep Alive



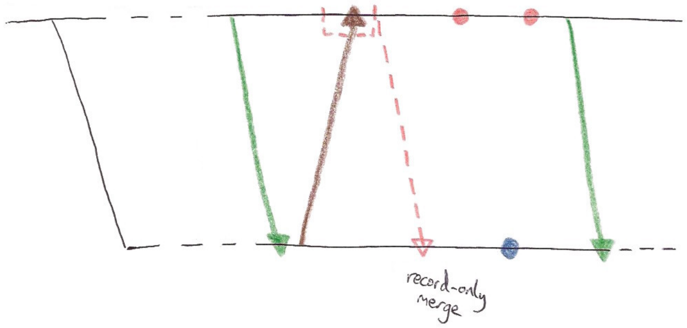
- Awkward extra step
- Doesn't work properly, in general

Keep Alive

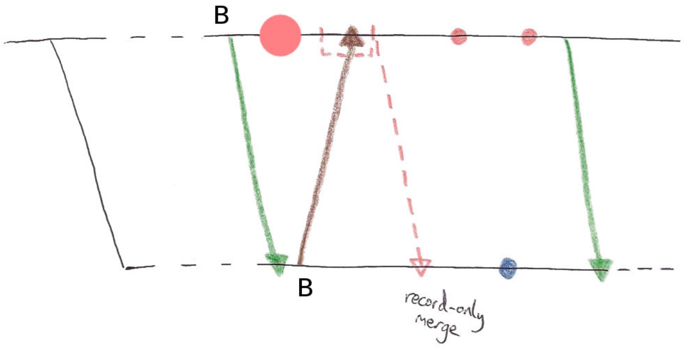


- Awkward extra step
- Doesn't work properly, in general

Keep Alive

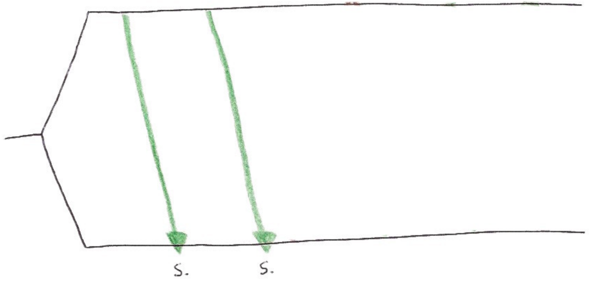


Keep-alive problem

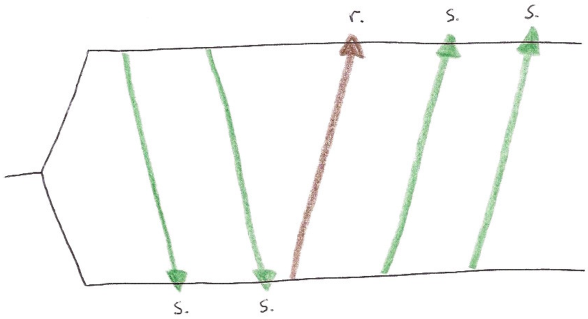


Outline

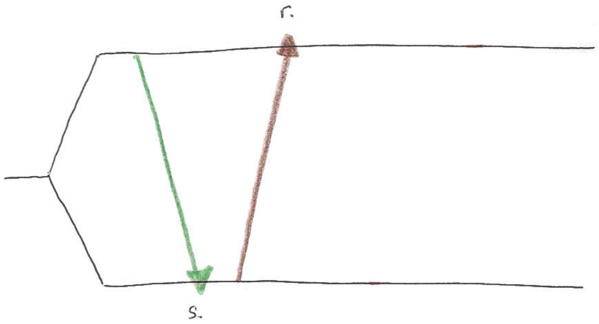
Merge the same way with *sync*



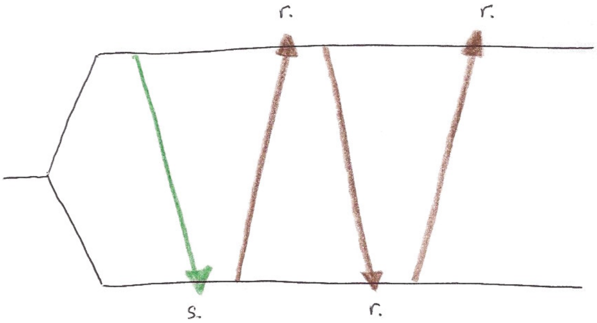
Merge the same way with *sync*



Merge the opposite way with *reintegrate*



Merge the opposite way with *reintegrate*



Surprise!
To-and-Fro Already Works

- Same direction again
  - sync
- Change direction
  - reintegrate

Surprise!
To-and-Fro Already Works

- Same direction again
  - sync
- Change direction
  - reintegrate

Surprise!
To-and-Fro Already Works

- Same direction again
  - sync
- Change direction
  - reintegrate

Surprise!
To-and-Fro Already Works

- Same direction again
  - sync
- Change direction
  - reintegrate

Outline

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run *"sync"* if base on source
  - Run *"reintegrate"* if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run “*sync*” if base on source
  - Run “*reintegrate*” if base on target

Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run "*sync*" if base on source
  - Run "*reintegrate*" if base on target

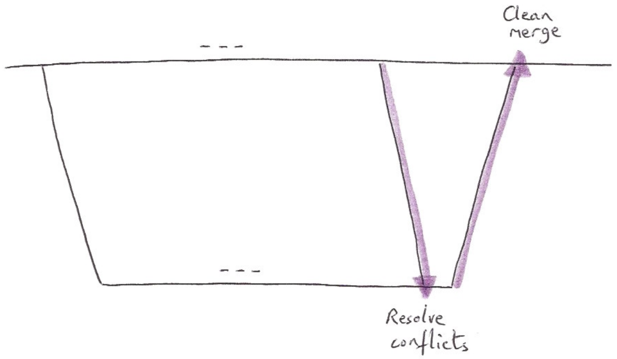
Algorithm

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Then, ideally...
  - Identify cherry-picks.
  - Break into 3-way merges, skipping the cherry-picks.
  - Perform the 3-way merges and mergeinfo addition.
- but currently...
  - Run “*sync*” if base on source
  - Run “*reintegrate*” if base on target

Limitations

- not yet symmetric inside
  - limitations NOT symmetric
  - results are symmetric
- *change-direction* merges
  - no cherry-picked revisions
  - no subtree-specific mergeinfo
  - no local mods in WC
  - no sparse WC
- in line with usage & best practice

Sync before reintegrating



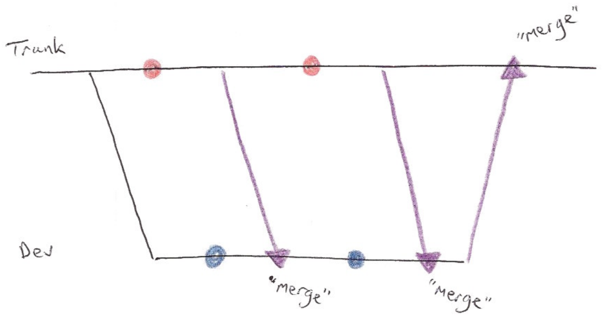
Outline

Leave out *–reintegrate*

```
+ svn merge --reintegrate ^/B A      # v1.7
--- Merging differences between repository URLs into 'A':
A    A/pickle
--- Recording mergeinfo for merge between repository URLs into
'A':
 U   A
```

```
+ svn merge ^/B A                    # v1.8
--- Merging differences between repository URLs into 'A':
A    A/pickle
--- Recording mergeinfo for merge between repository URLs into
'A':
 U   A
```

Use the same *merge* command



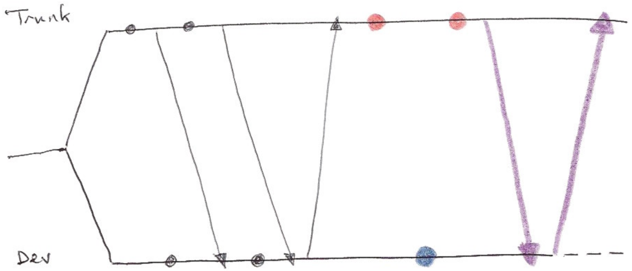
On-line Help for *svn merge*

```
$ svn help merge    # v1.7
merge: Merge changes into a working copy.
usage:
  1. merge SOURCE[@REV] [TARGET_WCPATH]
     (the “sync” merge)
  2. merge [-c M[,N...] | -r N:M ...] SOURCE...
     (the “cherry-pick” merge)
  3. merge --reintegrate SOURCE[@REV] [TARGET_WCPATH]
     (the “reintegrate” merge)
  4. merge SOURCE1[@N] SOURCE2[@M] ...
     (the “2-URL” merge)
...
```

On-line Help for *svn merge*

```
$ svn help merge    # v1.8
merge: Merge changes into a working copy.
usage:
  1. merge SOURCE[@REV] [TARGET_WCPATH]
     (the “automatic” merge)
  2. merge [-c M[,N...] | -r N:M ...] SOURCE...
     (the “cherry-pick” merge)
  3. merge SOURCE1[@N] SOURCE2[@M] ...
     (the “2-URL” merge)
...
```

Continue after reintegrating



Usability tweaks

Catch source/target mismatch

- source unrelated to target
- source same as target
- source is a subtree of target (or *vice-versa*)

Outline

History

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step

History

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step

History

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step

History

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step

History

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step

Rename Tracking Design

- Redesign
  - assume we'll be able to tell merge algo which src node matches which tgt node
- Modularize
  - a merge algorithm
  - a provider of rename info
  - a module to apply changes to WC
  - a mergeinfo read/write module
- Refactor
  - use merge logic for *merge*
  - use merge logic for *update* & *switch*
  - move merge logic to the server?

Rename Tracking Design

- Redesign
  - assume we'll be able to tell merge algo which src node matches which tgt node
- Modularize
  - a merge algorithm
  - a provider of rename info
  - a module to apply changes to WC
  - a mergeinfo read/write module
- Refactor
  - use merge logic for *merge*
  - use merge logic for *update* & *switch*
  - move merge logic to the server?

Rename Tracking Design

- Redesign
  - assume we'll be able to tell merge algo which src node matches which tgt node
- Modularize
  - a merge algorithm
  - a provider of rename info
  - a module to apply changes to WC
  - a mergeinfo read/write module
- Refactor
  - use merge logic for *merge*
  - use merge logic for *update* & *switch*
  - move merge logic to the server?

3-way Tree Merge

- in: rename tracking info
- responsible for
  - moves / renames
  - tree conflicts
- the rest (file merging) stays the same

Summary

- No more "–reintegrate": it's automatic
- To-and-Fro Merging
- Mergeinfo summary
- Sanity checks