Investigation of tree conflict handling in selected version control systems

Stefan Sperling

Free University of Berlin

November 13, 2008

Who am I?

- Stefan Sperling
- Studying at inf.fu-berlin.de since 2004
- Subversion developer since 2007
  Been working on Subversion's "tree-conflicts" branch and other areas.

Revision control basics

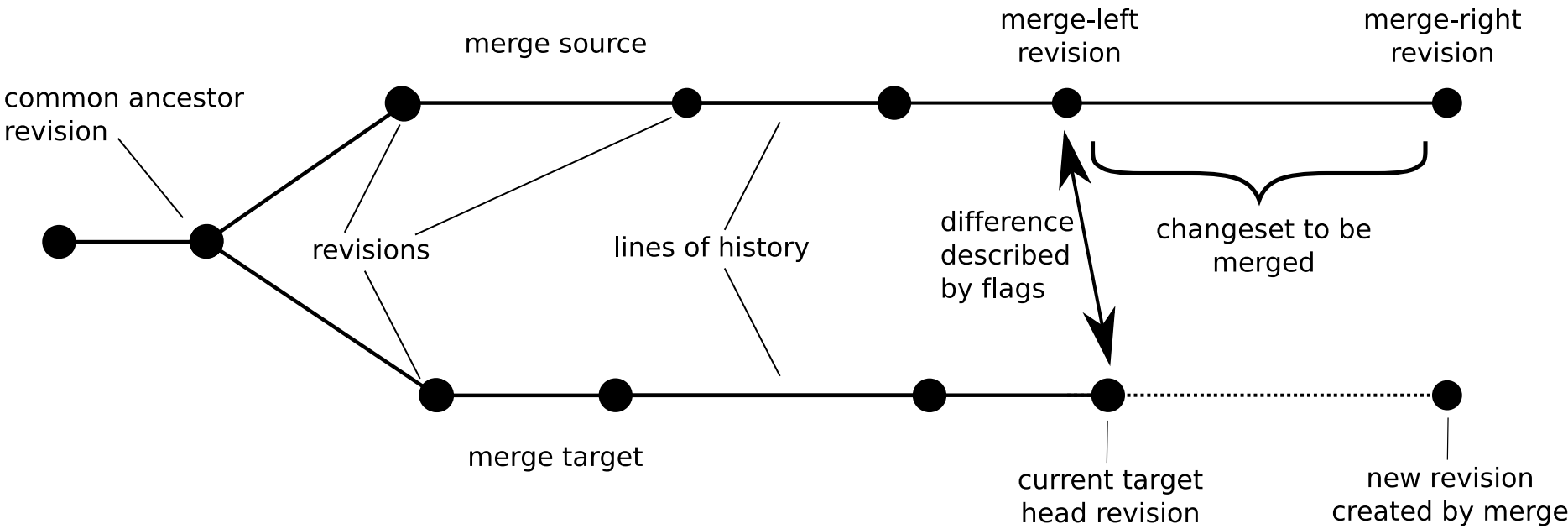
- repository
- working copy
- updating
- committing
- branching (parallel lines of development history)
  - feature branch
  - maintenance branch (e.g. release branch)
  - main line a.k.a. “trunk”
- merging (more details later)
- text conflict

What are tree conflicts?

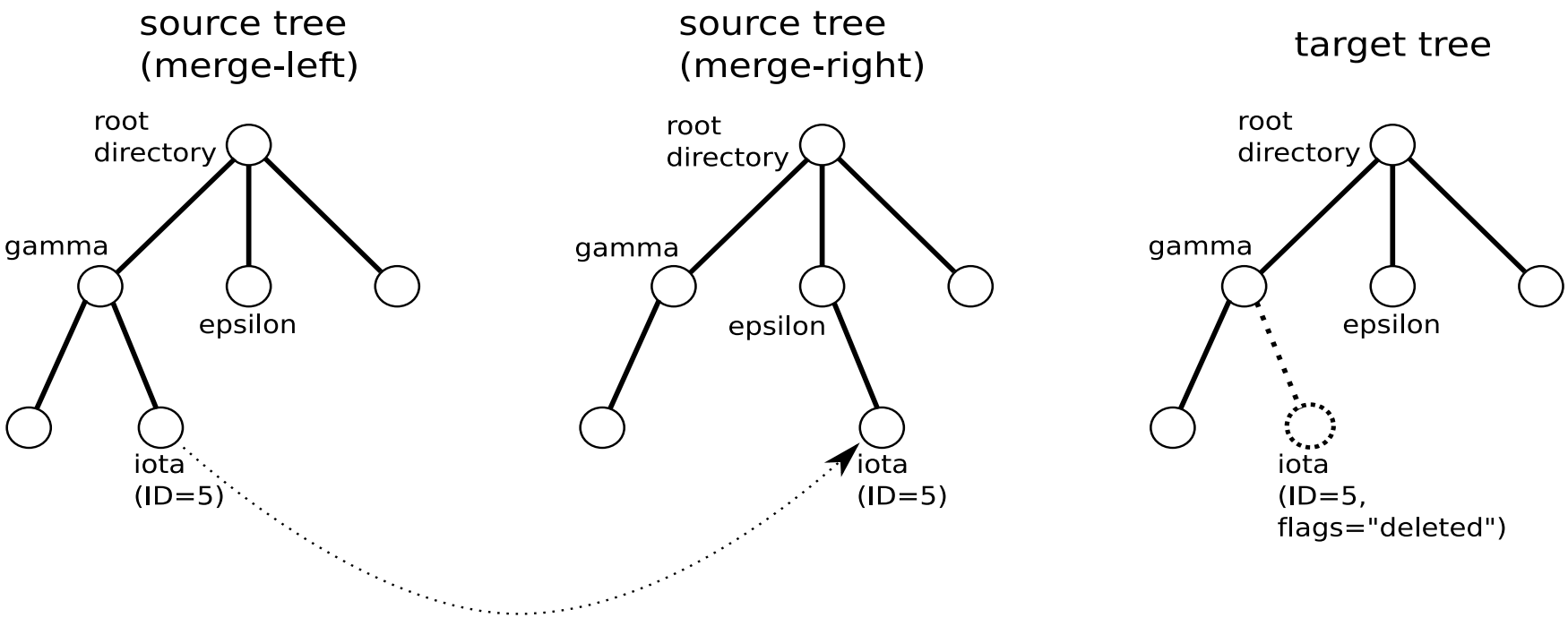
Merge conflicts at the level of directory tree structure:

- One side renames, the other deletes
- One side modifies, the other deletes
- One side modifies, the other renames
- One side deletes, the other copies
- ...

Merging


merge source
merge-left revision
merge-right revision
common ancestor revision
revisions
lines of history
difference described by flags
changeset to be merged
merge target
current target head revision
new revision created by merge

Example tree conflict scenario



Approach taken in the thesis

So this is what I did:

1. Model tree conflict scenarios in an *abstract* and *general* way
2. Reproduce scenarios with different version control tools (shell scripts)

Merge Scenarios (1)

	Operation			
Object state	Modify	Destroy	Copy (Phase 1)	Move (Phase 1)
object does not exist	**c**	m	**c**	**c**
no flags set	m	m	2	2
modified flag set	x	**c**	2	2
created flag set	-	-	-	-
destroyed flag set	**c**	m	**c**	**c**
moved flag set	m	**c**	2	**p**

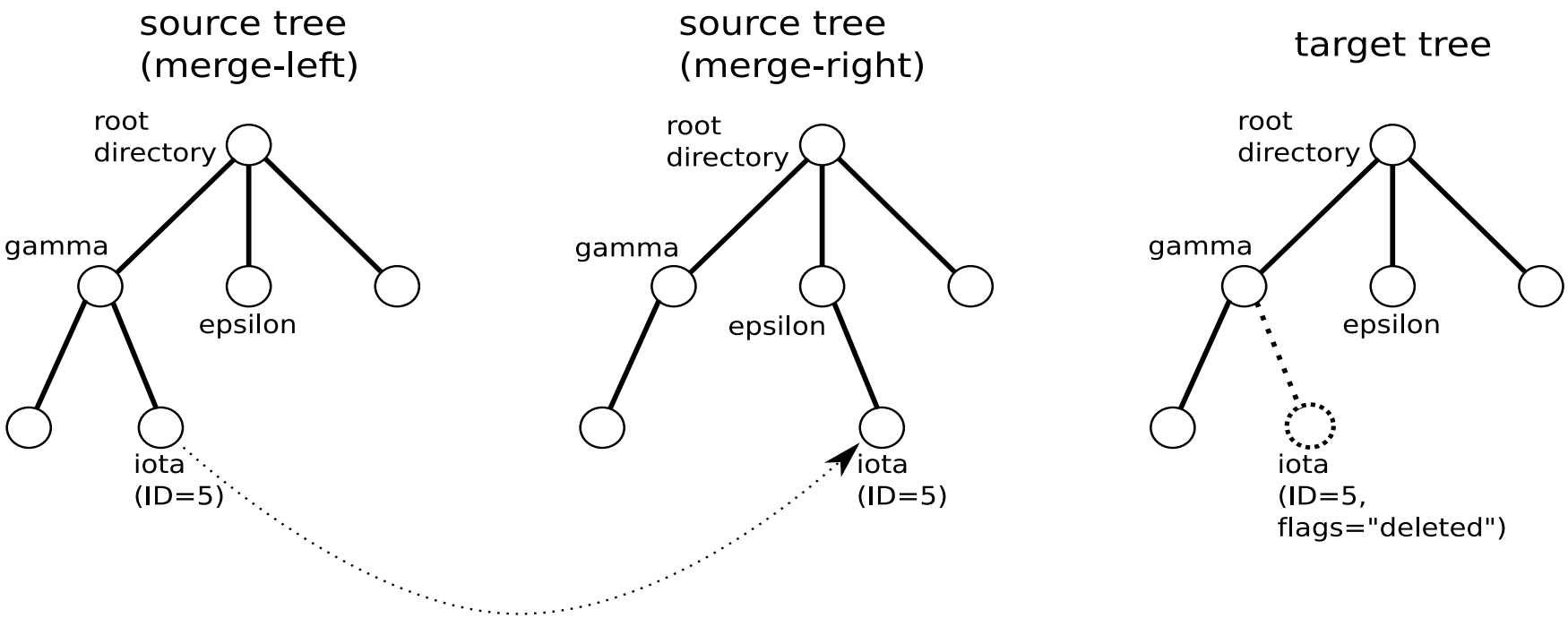
Table: Merge scenarios where operations look up objects by ID

Merge Scenarios (2)

	Operation		
Object state	Create	Copy (Phase 2)	Move (Phase 2)
object does not exist	m		
no flags set	-		
modified flag set	/		
created flag set	**c**		
destroyed flag set	m		
moved flag set	**c**		

Table: Merge Scenarios where operations look up objects by path

So what does this scenario look like in real life?



Mercurial:

```
# Move a file in repos
hg rename $repos/alpha $repos/alpha.moved
hg commit -m "moved alpha" $repos

# Delete the same file in repos2's working copy
hg remove $repos2/alpha

# Pull the modification into repos2 and update
(cd $repos2 && hg pull; hg update)
```

Mercurial does not detect the conflict

The file renamed in the source branch is:

1. Created in the target branch at its new location
2. Left destroyed in the target branch at its old location

This merge result can be committed without warning!

```
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ ls
alpha.moved  beta         epsilon/     gamma/
```

Git

```
# Move a file in repos
(cd $repos && git mv alpha alpha.moved)
(cd $repos && git commit -a -m "moved alpha")

# Remove the same file in repos2.
(cd $repos2 && git rm alpha)
# We need to commit here, else git won't detect the conflict
(cd $repos2 && git commit -a -m "destroyed alpha")

# Pull the move into repos2 and merge
(cd $repos2 && git pull)
```

Git detects the conflict

The file renamed in the source branch is:

1. Created in the target branch at its new location
2. Left destroyed in the target branch at its old location

But there is a warning:

```
   24dc595..6b4e285  master     -> origin/master
CONFLICT (rename/delete): Renamed alpha->alpha.moved in
6b4e2855a9b66dfde4350c8815c6e2050ffd6301 and deleted in HEAD
Automatic merge failed; fix conflicts and then commit the result.
$ ls
alpha.moved  beta         epsilon/     gamma/
```

Subversion

```
# Move a file in trunk
svn move $trunk/alpha $trunk/alpha.moved
svn commit -m "moved alpha" $trunk

# Destroy the same file in the branch
svn remove $branch/alpha
# Committing here does not affect the merge result

# Merge the copy into the branch
svn merge $trunk_url $branch
```

Subversion ignores the conflict

The file renamed in the source branch is:

1. Created in the target branch at its new location
2. Skipped in the target branch at its old location

```
--- Merging r2 through r3 into '.':
A    alpha.moved
Skipped missing target: 'alpha'
$ ls
alpha.moved  beta          epsilon/     gamma/
```

Results (files)

Operation	**Object state**	Mercurial 1.0.1	Git 1.5.6.4	Subversion 1.5.1
Modify	no object	/	**c**	-
Modify	destroyed	**c**	**c**	-
Create	created	x	x	x
Create	moved	x	**c**	x
Destroy	modified	**c**	**c**	-
Destroy	moved	-	**c**	-
Copy phase 1	no object	/	-	-
Copy phase 1	destroyed	-	-	-
Copy phase 2	created	x	x	x
Copy phase 2	moved	x	**c**	f
Move phase 1	no object	/	**c**	-
Move phase 1	destroyed	-	**c**	-
Move phase 1	moved	**c**	**c**	-
Move phase 2	created	x	**c**	x
Move phase 2	moved	x	**c**	f

Results (directories)

Operation	**Object state**	Mercurial 1.0.1	Git 1.5.6.4	Subversion 1.5.1
Create	created	-	-	-
Create	moved	-	-	-
Destroy	modified	-	-	-
Destroy	moved	-	**C**	-
Copy phase 1	no object	/	-	-
Copy phase 1	destroyed	-	-	-
Copy phase 2	created	-	-	-
Copy phase 2	moved	-	-	f
Move phase 1	no object	/	**C**	-
Move phase 1	destroyed	-	**C**	-
Move phase 1	moved	**C**	**C**	-
Move phase 2	created	-	-	-
Move phase 2	moved	-	-	f

Gero

Du musst nach dem Vortrag Gero aus der ZEDAT abholen und mit ihm ins Kauderwelsch gehen!

That was it!

Thank you!
Questions?