Convergence vs. Divergence

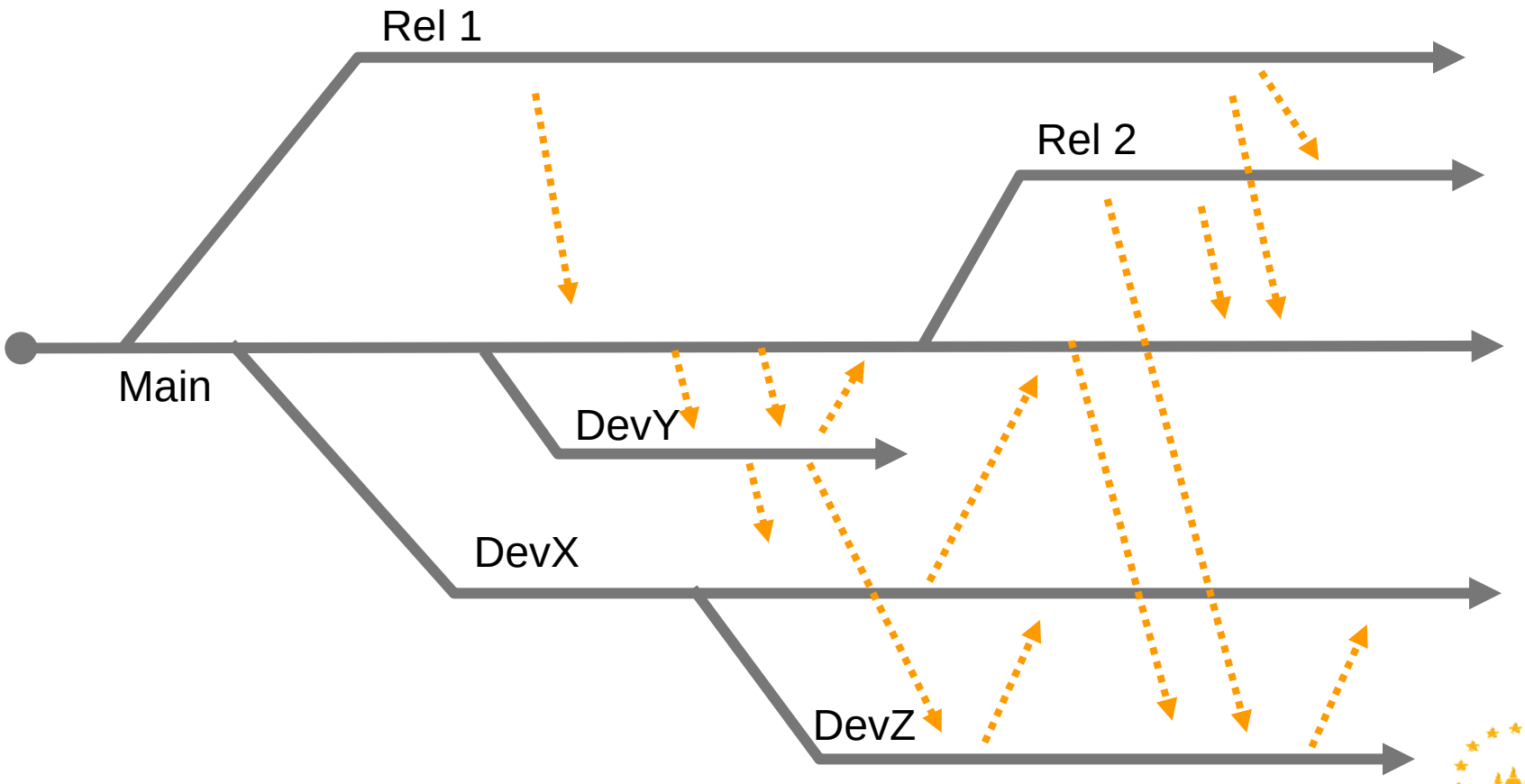
Purposeful Merging with Perforce

Laura Wingerd • Perforce Software • www.perforce.com

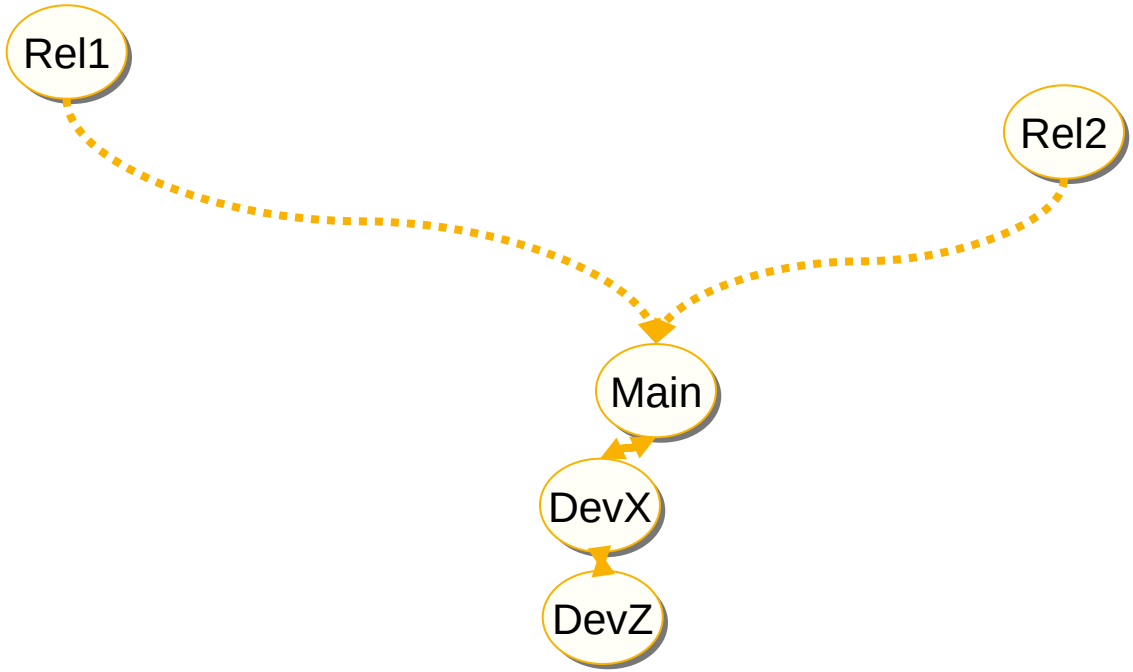
Overview

- Background
  - Branching and merging
  - 3-way file merging
- Convergence vs. divergence
  - Some branches should converge, some should diverge
  - Merging for convergence is not the same as merging for divergence
- Purposeful merging
  - How to get the merge result you want

Branching...



Branches over time...



- Some branches converge, some branches diverge

Branches diverge when…

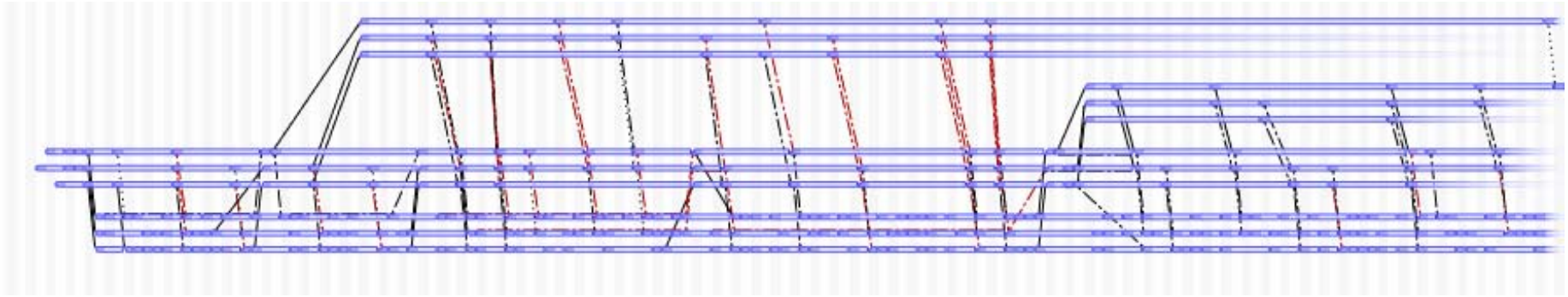
- back-porting to release lines
  - release lines always diverge from trunk
- cherry-picking
  - e.g., hunt-and-peck feature packaging
- branching for customization
  - e.g., per-customer, per-platform, etc.
- trading changes between developers
  - e.g., boilerplates, code snippets

Branches converge when...

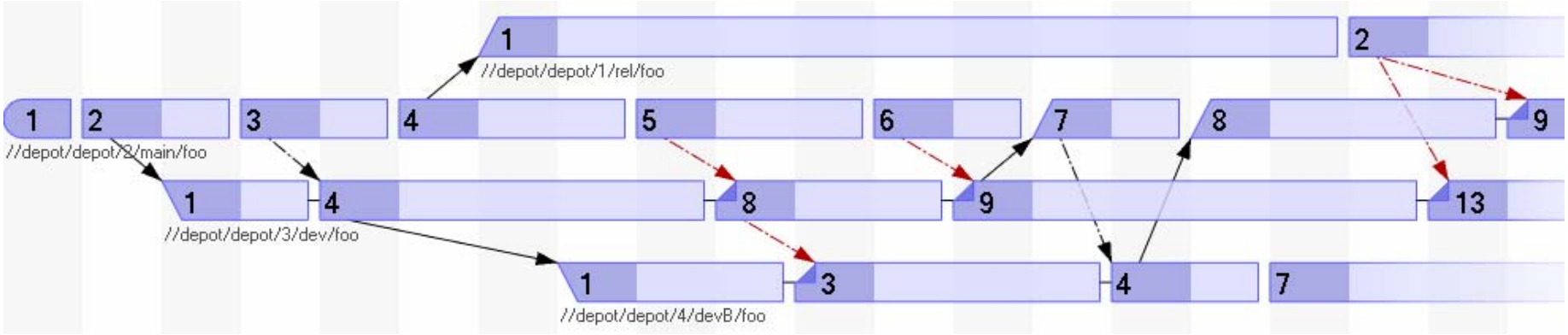
- development tasks are completed
  - i.e., delivering completed work from dev line to mainline
- isolated development projects are rejoined
  - delivering private branch work into a shared branch
- distributed development is reconciled
  - developers working in separate repositories

Merging files

- Although we speak of “merging branches”...
- ...merging takes place between individual pairs of *source* and *target* files



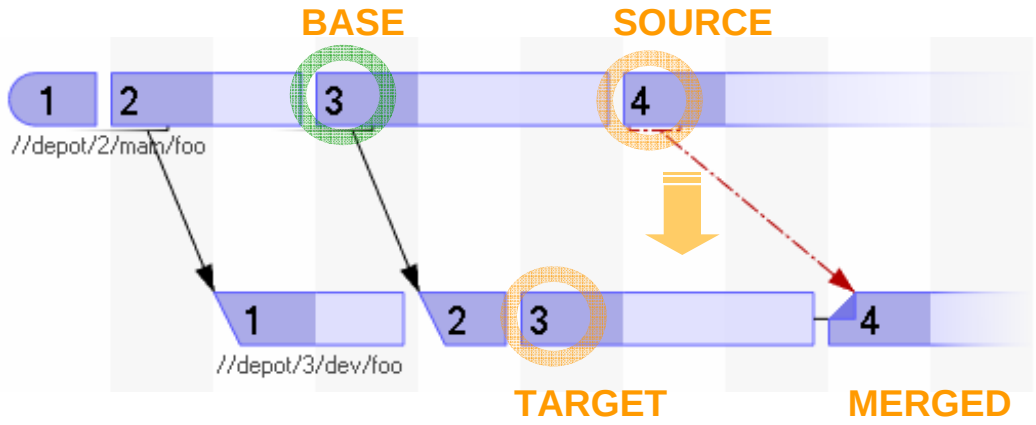
Integration history “arrows”



- Integration history records source-to-target relationships between file revisions
- Perforce uses integration history to determine:
  - which source revisions still need to be integrated to target
  - the revision that will be used as the base for a three-way merge from source to target

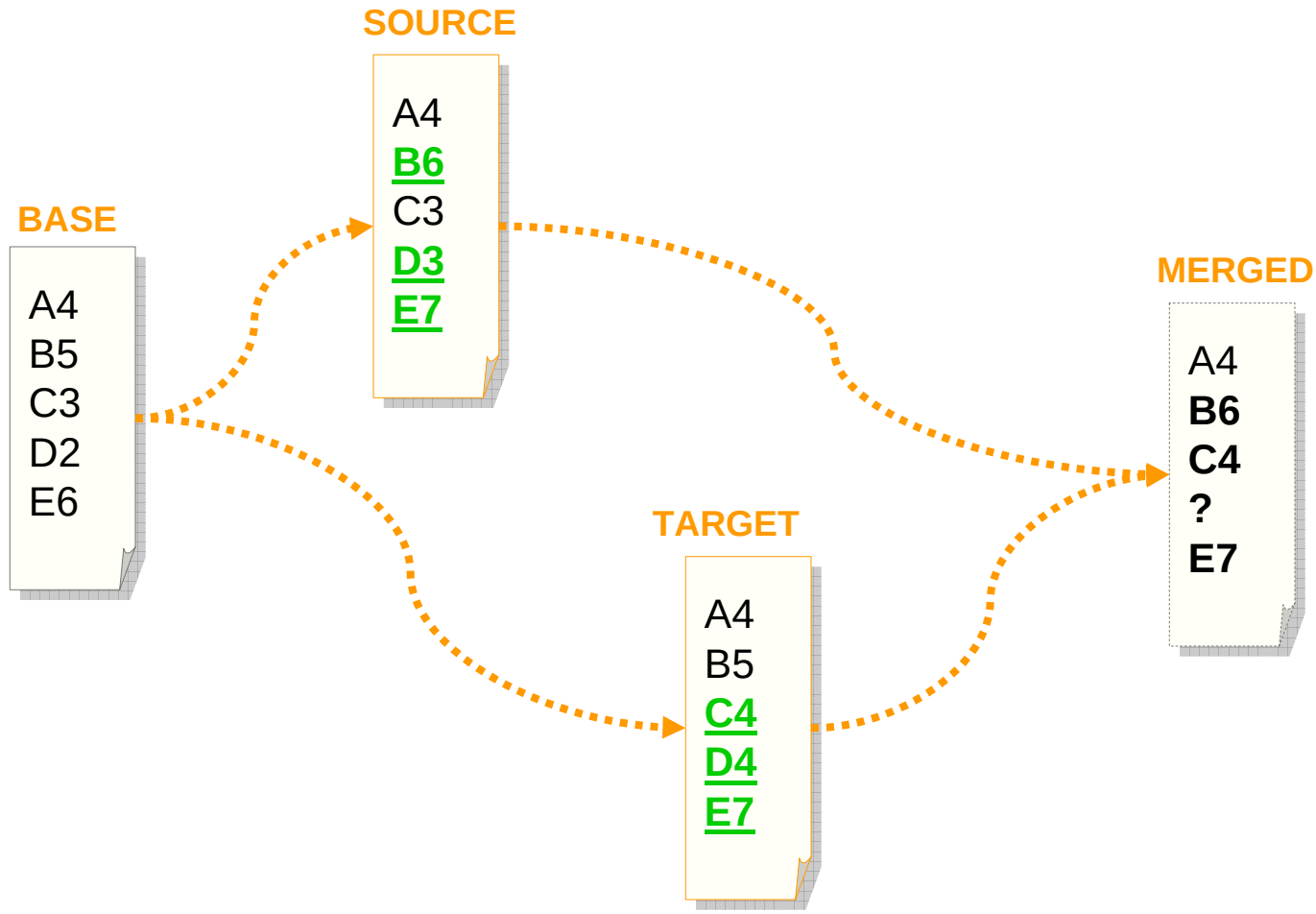
Three-way file merging

- A *source* file version and *target* file version are merged with reference to a *base* file version



- The merged result is a consequence of:
  - merge tool capabilities, and...
  - base selection

The essence of a three-way merge



Three-way file merge tools vary...

- File formats

- Granularity

```
[JavaScript Prompt Dialog]
Title = 19008
Group, 0,
MultilineEdit, 0, Description_label,
Edit,  0,Text_edit,
```

- Syntax awareness

```
<figure id="create_gizmo">
    <title>
        The 'Create Gizmo' dialog
    </title>
    <graphic fileref="img/cg.gif"/>
</figure>
```

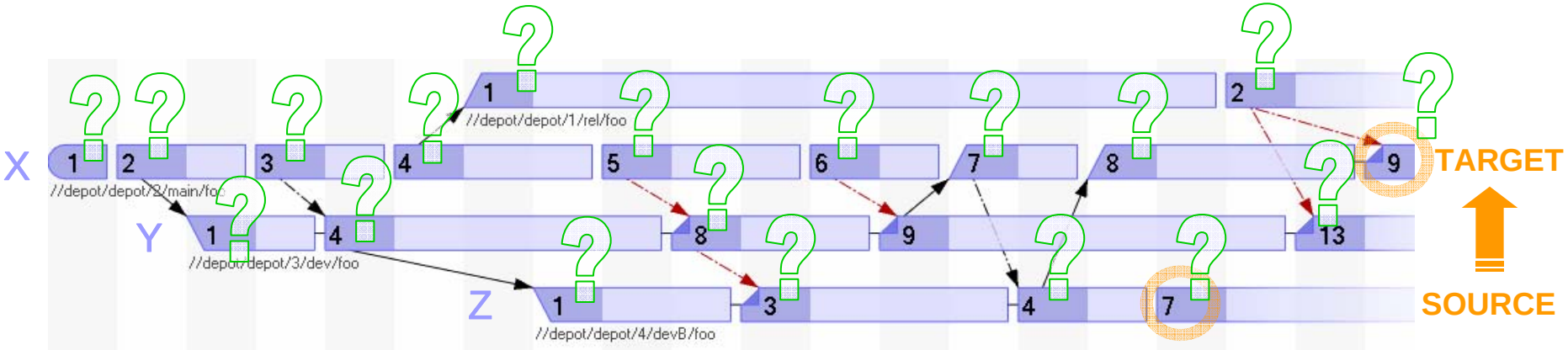
- Conflict resolution

Perforce's merge tools

- File formats: text 
- Syntax awareness: none
- Granularity: line
- Conflict resolution: "diligent"
  - Does more auto-resolving
  - Leaves fewer conflicts to be resolved manually

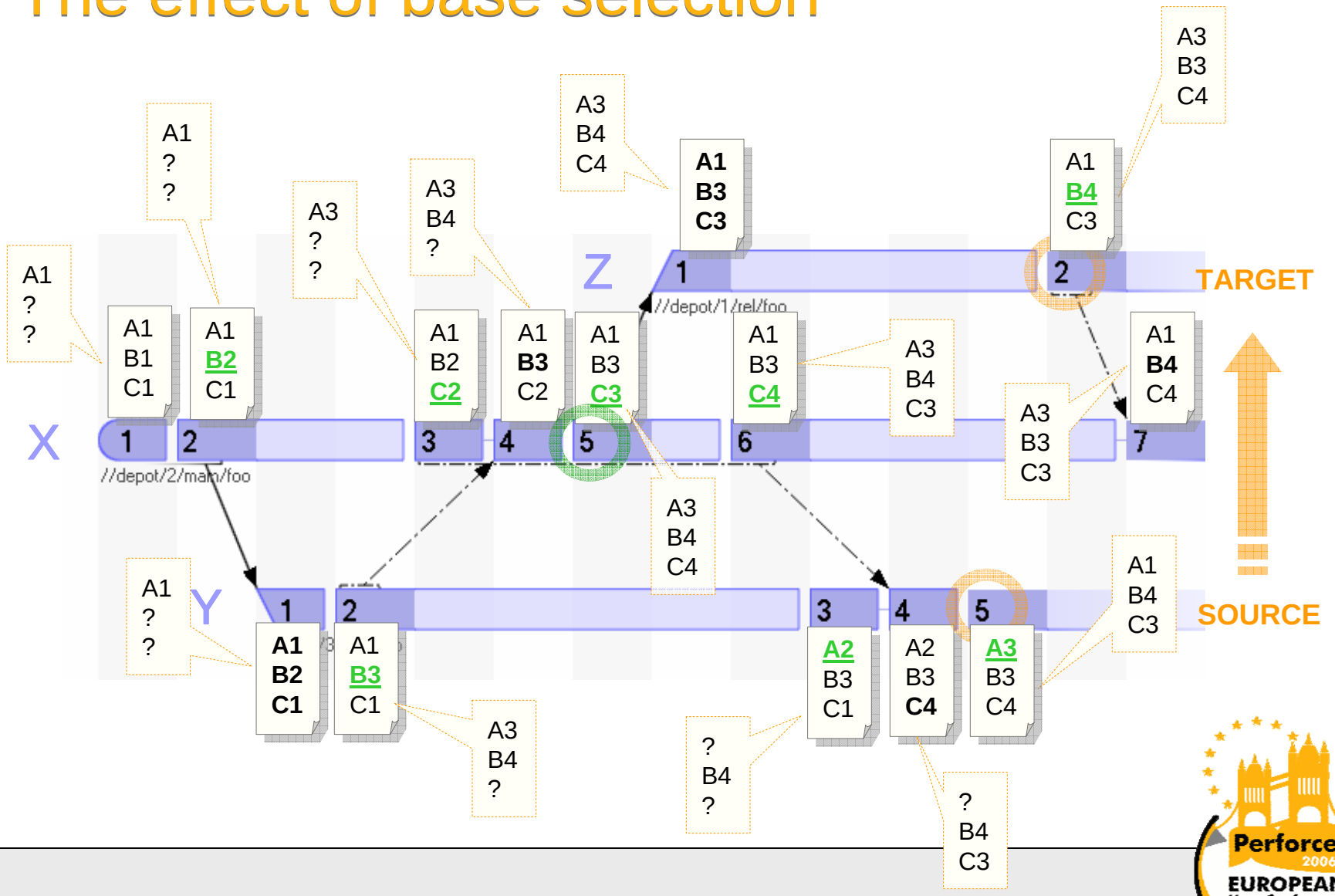
```
<figure id="create_gizmo">
    <title>
        The 'Create Gizmo' dialog
    </title>
    <graphic fileref="img/cg.gif"/>
</figure>
```

What makes a good merge base?

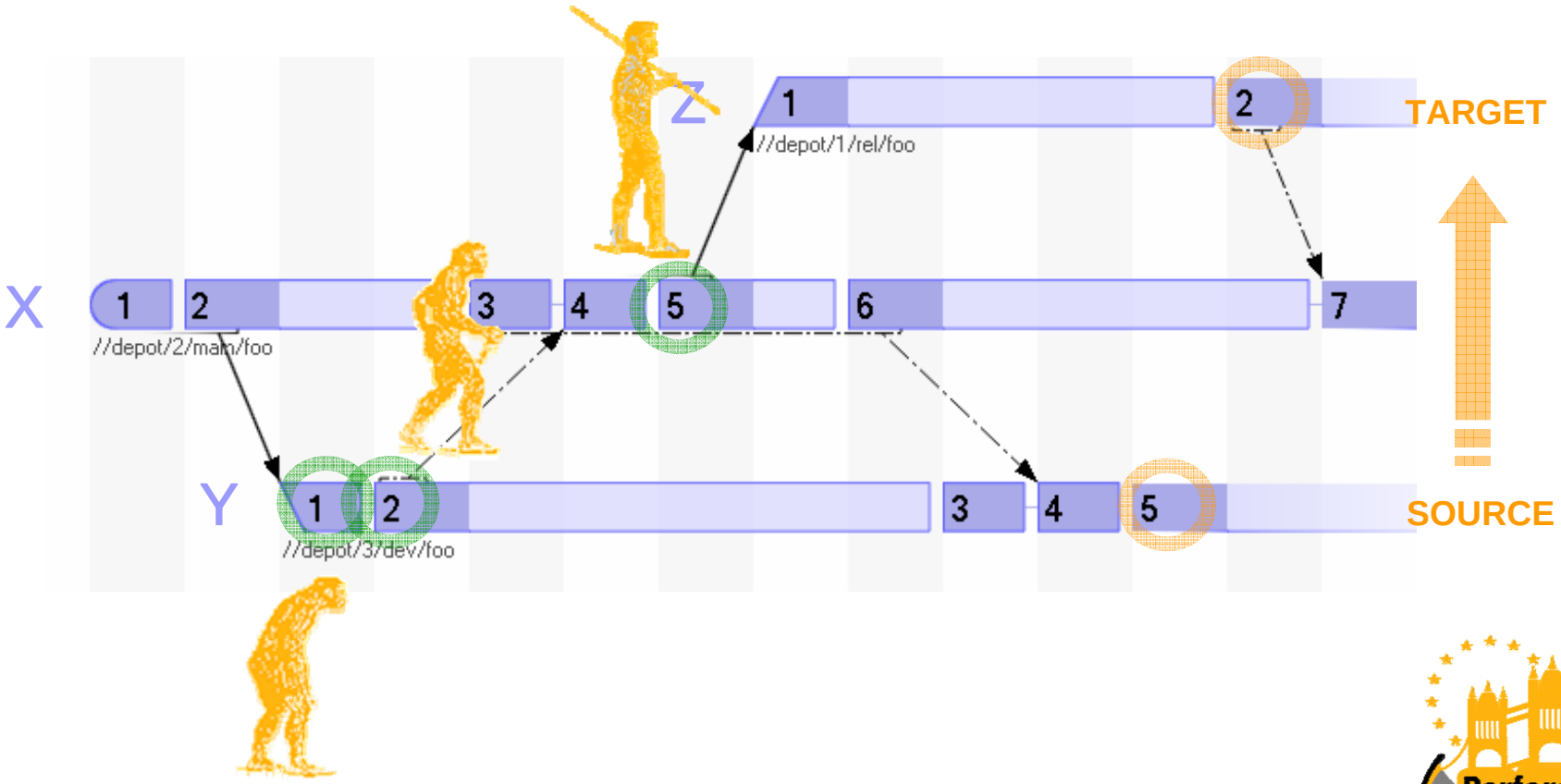


- A base with “qualifying history” – i.e., whose complete history is in the history of both source and target...
- ... and a base with *lots* of history – i.e., whose history has as much in common with both the source’s history and the target’s history as possible

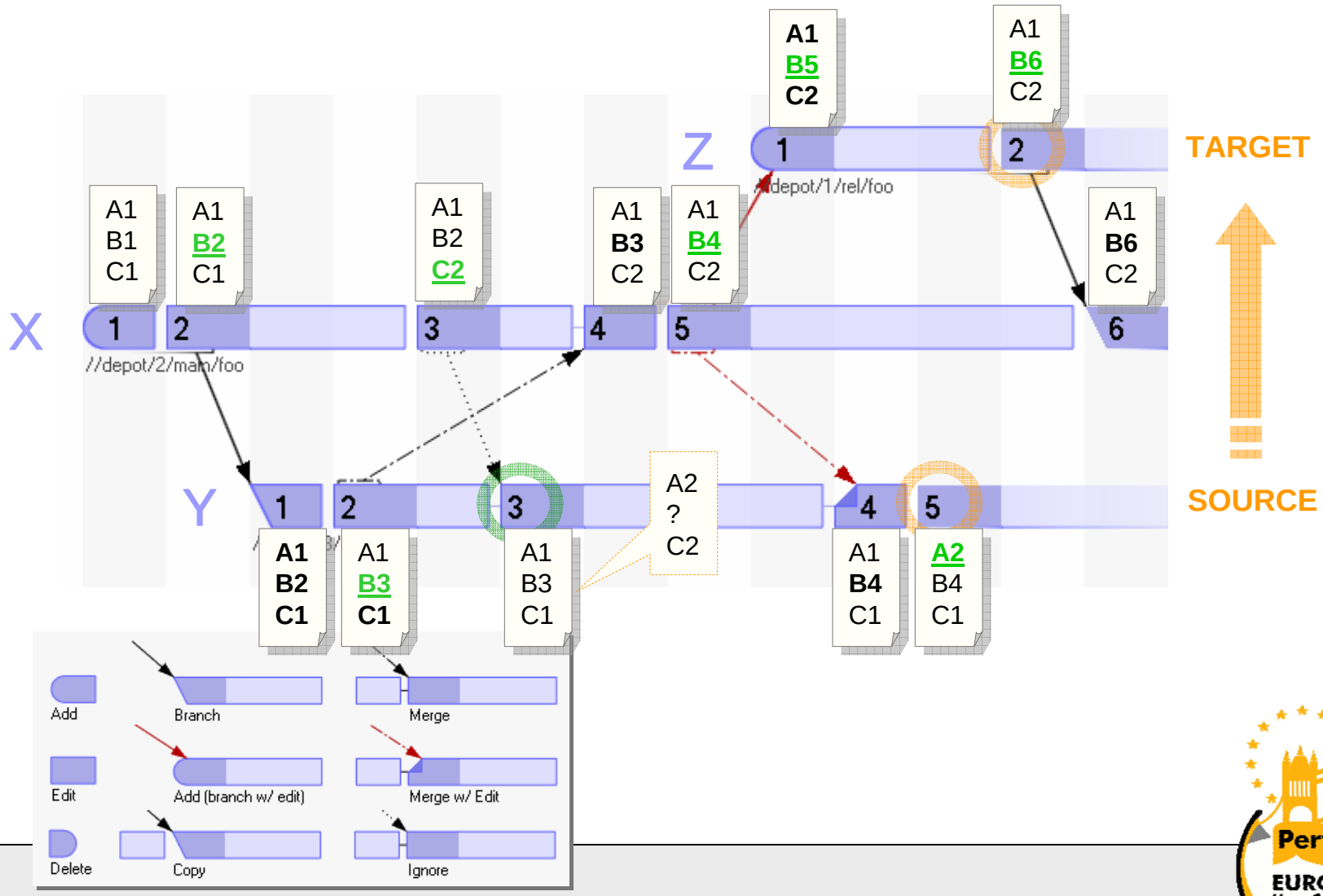
The effect of base selection



Base selection through the ages

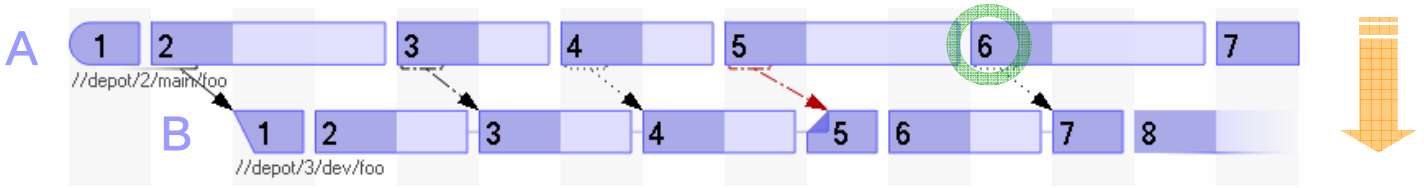


Arrow types and base selection

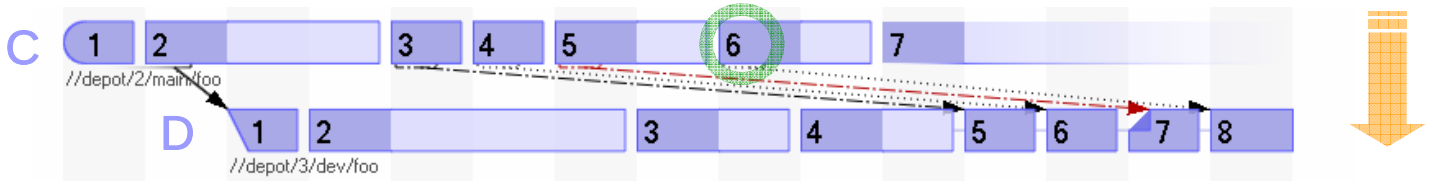


Preserving divergence

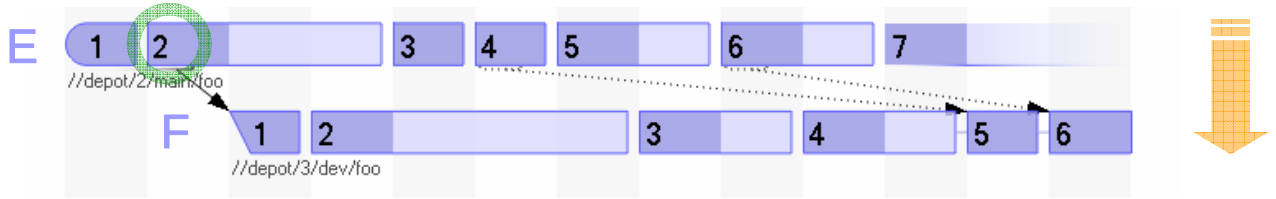
- Continuous, incremental merging preserves divergence



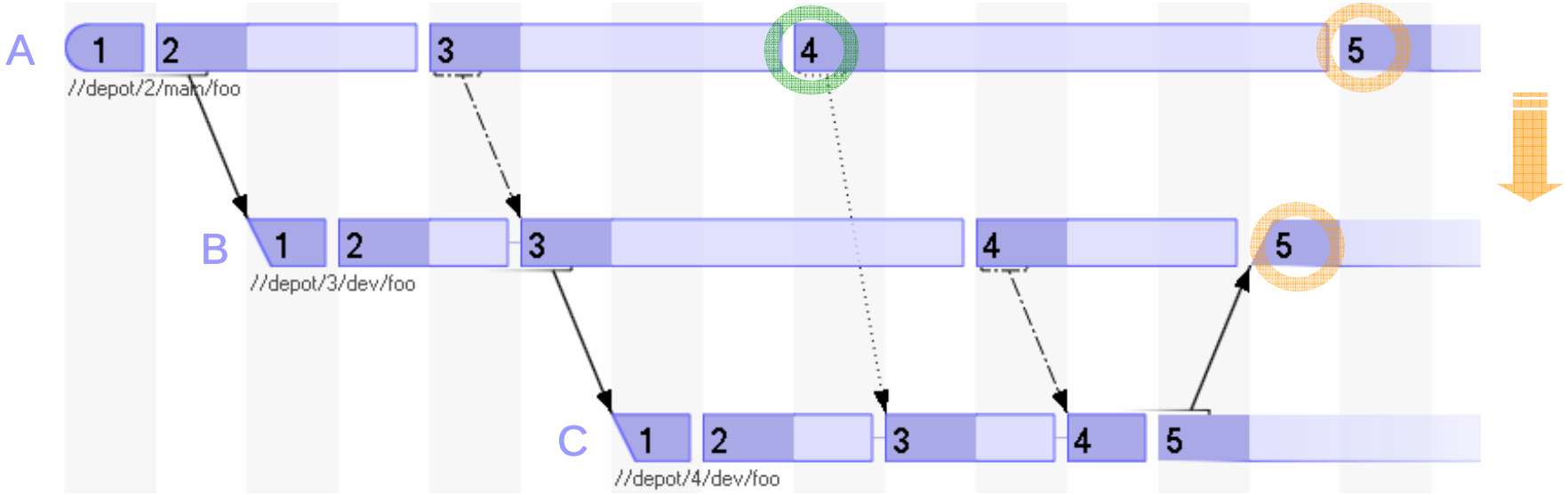
- After-the-fact, incremental merging preserves divergence



- Cherry-picking before mass merging does *not* preserve divergence

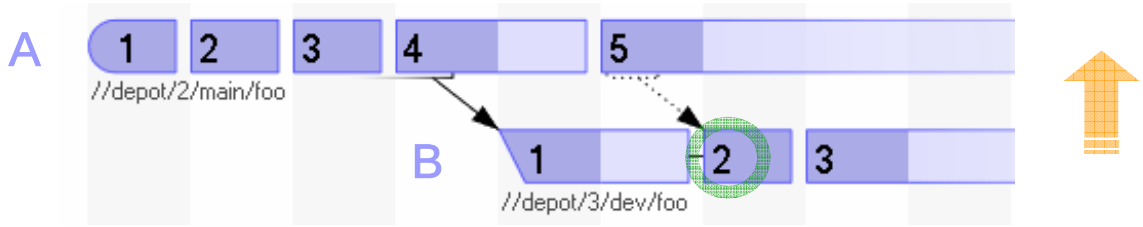


“Inherited” divergence

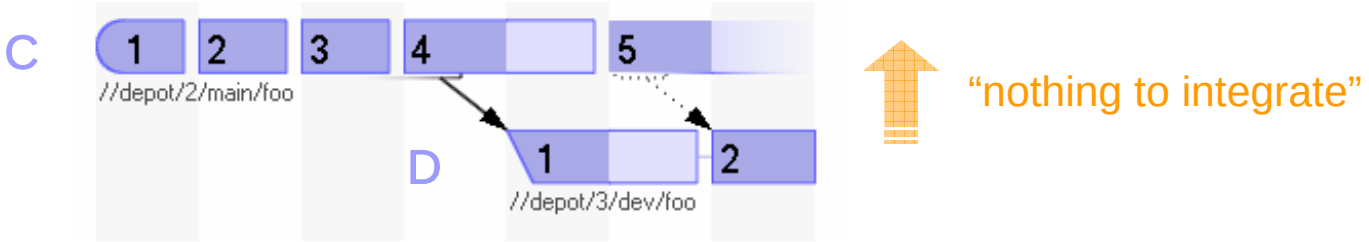


Unintentional divergence

- An “ignore” arrow is not the same as backing out a change

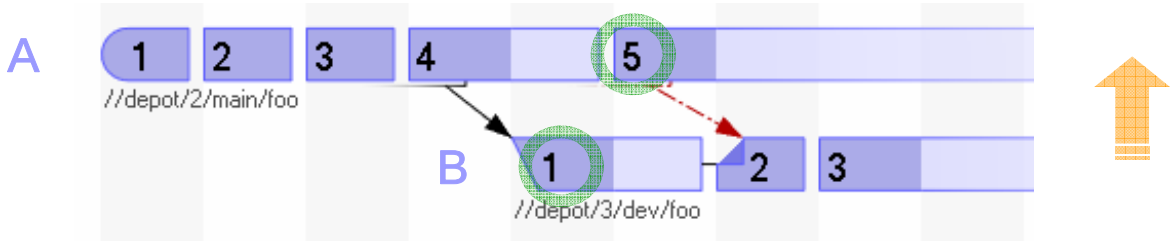


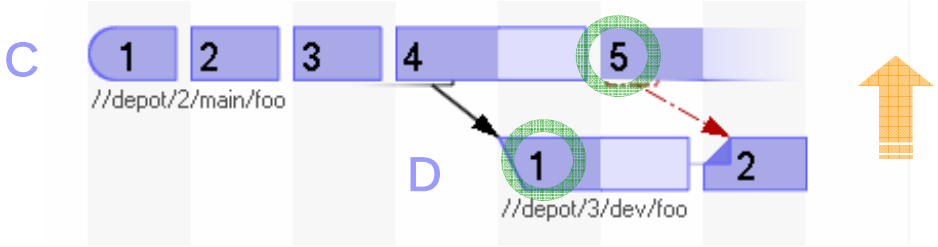
- “Merge down, copy up” can be foiled by “ignore” arrows



The effect of “edit” arrows

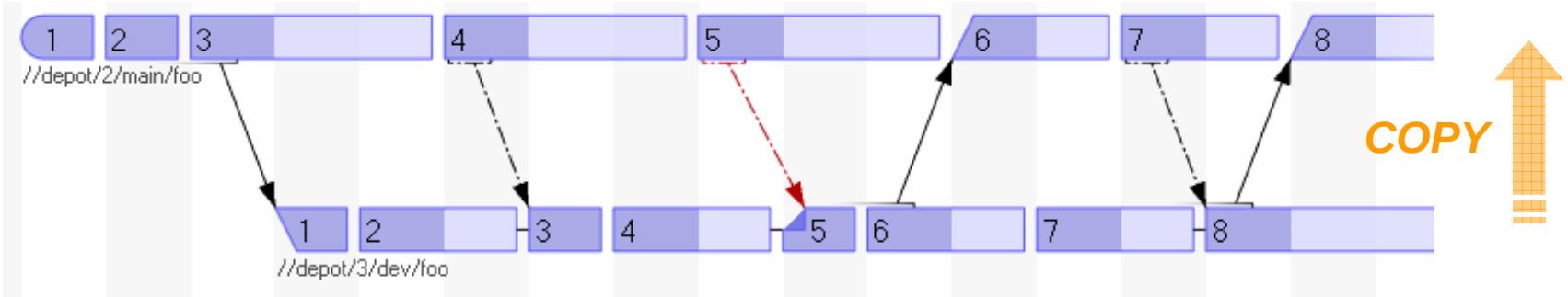
- Change propagation is assured, but convergence is not guaranteed





Guaranteeing convergence

- “Merge down, copy up”



Assuring a correct copy

- The recipe:

```
1.  p4 integ –n [target] [source]
2.  p4 integ –f [source] [target]
3.  p4 resolve –at
4.  p4 diff –sr | p4 –x - revert
5.  p4 integ [source] [target]
6.  p4 resolve –at
7.  p4 submit
```

In a nutshell:

- Some branches diverge, some branches converge.
- With three-way merging, base selection determines divergence or convergence.
- As of Rel 2006.1, Perforce's base selection accommodates both convergence and divergence nicely, but is biased toward preserving divergence
- The "merge down, copy up" method achieves convergence between branches – as long as you're copying branches correctly.

Convergence vs. Divergence

Purposeful Merging with Perforce

Laura Wingerd • Perforce Software • www.perforce.com