Discussion questions for *A Principled Approach to Version Control*

COMP 150 - Applied Functional Programming

November 14, 2012

Analyzing the technical development

(1) On the second page, under **Terminology**, the authors give this model of a repository:

> In addition to this current representation of its state, a repository contains *history*. Each repository's internal state is the result of several *patches*. A patch is a logically connected collection of operations that changes the state of a repository.

Is this model consistent with git? How would you relate this model to our model of a git repository?

(2) Please sort the various bits of formalism in Sections 2 and 3 into the following four rough-and-ready categories:

- Language
- Denotational semantics
- Algebraic laws
- Proofs of soundness

(3) Using equational reasoning, please give a the proof of Corollary 5. It might help you to know that $\triangle$ is an associative and commutative operator, that the empty set is the identity of $\triangle$, and that for any set X, $X \triangle X = \emptyset$.

(4) How important is Corollary 11 in Section 5.1? And what's missing from this development?

(5) Section 5.2 defines the notions of *pull* and *minimal pull*. How does this definition compare with your mental model of what git does? What about darcs?

Analyzing the research contributions

(6) Starting in Section 2.1 and continuing in Section 4, the paper develops a richer and richer model of *repository state.* Informally,

- What is the model?
- What are the advantages of the model?
- What are the drawbacks of the model?

(7) In the concluding section, the authors write

> Our theory allows a represention of text files in such a way that formally, merging is completely superfluous. When moving patches between repositories, the important question is not how to merge, but only whether the resulting repository is consistent.

Sounds like "problem solved," doesn't it? What work remains to be done?

(8) What is reasonable to expect from a paper that presents a theory of software? What is reasonable to expect from a paper on the theory of source-code control? Why do we suppose this paper has not been published?

(9) On the front page, the authors say that

> Programmers dread performing complex operations on a repository, such as merging branches or resolving conflicts. These operations can be both unwieldy and unpredictable.

How are these problems addressed in the paper?

Relating the contributions to our prior work

(10) How do you relate the material in the paper to the theoretical work we have done together in class?

(11) What issues are addressed in the paper that we have not covered in class?

(12) What issues have we addressed in class that are not covered in the paper?

Bonus question

(13) Section 4.3 proposes to deal with line-based text files by giving each line a label. This technique is also used by the collaborative editor in the Treedoc paper (which we have not discussed in class). For source-code control, what are the advantages and disadvantages of this representation?