University of Bergen
Department of informatics

A Very Good Title

Åsmund Aqissiaq Arild Kløvstad

Supervised by Håkon Robbestad Gylterud

UNIVERSITY OF BERGEN

Faculty of Mathematics and Natural Sciences

January, 2022

Abstract

Lorem ipsum dolor sit amet, his veri singulis necessitatibus ad. Nec insolens periculis ex. Te pro purto eros error, nec alia graeci placerat cu. Hinc volutpat similique no qui, ad labitur mentitum democritum sea. Sale inimicus te eum.

No eros nemore impedit his, per at salutandi eloquentiam, ea semper euismod meliore sea. Mutat scaevola cotidieque cu mel. Eum an convenire tractatos, ei duo nulla molestie, quis hendrerit et vix. In aliquam intellegam philosophia sea. At quo bonorum adipisci. Eros labitur deleniti ius in, sonet congue ius at, pro suas meis habeo no.

Acknowledgements

Est suavitate gubergren referrentur an, ex mea dolor eloquentiam, novum ludus suscipit in nec. Ea mea essent prompta constituam, has ut novum prodesset vulputate. Ad noster electram pri, nec sint accusamus dissentias at. Est ad laoreet fierent invidunt, ut per assueverit conclusionemque. An electram efficiendi mea.

Åsmund Aqissiaq Arild Kløvstad
Tuesday 18th January, 2022

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Background

In this section we summarize some key papers and their significance to the project.

We may also include a brief introduction to HoTT, but maybe that's better suited to 2.0.5.

1.1.1 A Categorical Theory of Patches

A Categorical Theory of Patches [8] defines a category of files and patches, such that a merge is a pushout. To ensure a merge is always possible they first construct the category $\mathcal{L}$ of files and patches, and then its conservative cocompletion $\mathcal{P}$.

$\mathcal{P}$ contains all finite colimits – and in particular all pushouts – so the merge of a span is always defined. The paper's chief achievement is the explicit construction of this category and these pushouts.

Interesting insights I'm not sure how to incorporate:

- the construction of $\mathcal{P}$ can be understood as the addition of *partially* ordered files to $\mathcal{L}$.

- "flattening" these partial orders leads to cyclic graphs. On editing text [4] objects, but maybe not correctly
- the poset structure of $\mathcal{L}$ and $\mathcal{P}$ is given explicitly by $\mathcal{G}$ and the nerve functor $N_{\_}$ (!!).

(maybe mention Pijul [1] (if so, figure out the relationship to [8])) (maybe some figures go here)

1.1.2 Patch Theory (Darcs)

Here we discuss several proposed formalisms for a the patch theory employed by Darcs [2]. [7, 12, 5] all attempt to describe Darcs' patch theory. (focus on Lynagh, I think)

Lynagh [7] proposes an "algebra of patches" as a theoretical basis for the Darcs [2] version control system.

In this model a repository state is a set of updates (called *patches*, but we want to avoid that ambiguity) and a patch is a change to this set. For example pulling the repository $\{c\}$ into the repository $\{a, b\}$ results in a new repository $\{a, b\} \cup \{c\} = \{a, b, c\}$.

Patches are only applicable to one repository state, and result in a new state. If they are compatible, we may string them together into a *patch sequence*. Denoting the previous example patch by P and the "do-nothing" patch by Id we have $\{a, b\}P\{a, b, c\}Id\{a, b, c\}$ – pulling $\{c\}$ followed by doing nothing. The repository state may be omitted from sequences.

Finally a notion of *commutation* of patches is defined. We say patches A and B commute if the span $\bullet \xleftarrow{A} \bullet \xrightarrow{B} \bullet$ can be expanded to the following commuting square:

$$\begin{tikzcd} \bullet \arrow[r, "B"] \arrow[d, "A"'] & \bullet \arrow[d, dashed, "A'"] \\ \bullet \arrow[r, dashed, "B'"'] & \bullet \end{tikzcd}$$

and write $AB \leftrightarrow B'A'$.

We have four axioms for patches and commutation:

1. Commutativity (lol) (3.1): $AB \leftrightarrow B'A' \iff B'A' \leftrightarrow AB$
2. Invertibility (3.2): for each A there is an A^{-1} s.t $AA^{-1} = A^{-1}A = Id$
3. Inv-cong (3.3): $AB \leftrightarrow B'A' \iff A^{-1}B' \leftrightarrow BA'^{-1}$
4. Circular (3.5/6): performing all pairwise commutations in a sequence gets us back to the beginning (or, a horrible equation)

With these axioms and definitions we can merge the span $\bullet \xleftarrow{A} \bullet \xrightarrow{B} \bullet$ by filling in the square:

$$\begin{tikzcd}
\bullet \arrow[r, "B"] \arrow[d, bend right, "A"'] & \bullet \\
\bullet \arrow[u, dashed, bend right, "A^{-1}"'] \arrow[r, dashed, "B'"'] & \bullet \arrow[u, dashed, "A^{-1\prime}"']
\end{tikzcd}$$

We have $A^{-1}B \leftrightarrow B'A^{-1\prime}$ and define $merge A\ B = AB'$. It is not clear to me from this diagram why the inverse(s) is needed

1.1.3 Homotopical Patch Theory

Homotopical Patch Theory [3] gives a formulation of patch theory in homotopy type theory. A patch theory is represented by a higher inductive type, and its interpretation by a function out of this type.

By representing repository state as points and patches as paths in a higher inductive type, the groupoid structure of the patch theory comes “for free”. Paths come with composition, and by the groupoid laws this composition is associative, unital, and respects inverses. Additionally, functions (which are functors) respect this structure so any interpretation must also validate the groupoid laws.

Patch laws are represented by paths between paths (squares? disks? 2D-somethings). For example we may want the application of two independent patches to commute – this is done with a patch law.

While the HIT formulation gives a lot “for free”, it also has some drawbacks. In particular, the requirement that all patches have inverses causes some problems. The workaround is to “type” patches with the history they are applicable to. This allows Angiuli et al. to define a merge operation in terms on only the “forward” patches, but leads to a fairly complex theory even for relatively simple settings.

An interesting feature of Angiuli et al.'s patch theories is that the type of repositories must be contractible. Since patches are represented by paths, any point can be retracted along them. As such, all repositories are – in a sense – “the same” and we need better notions of “sub-homotopical” [3] computations to reason about their differences.

1.1.4 Path Spaces of Higher Inductive Types

Path Spaces of Higher Inductive Types in Homotopy Type Theory [6] provides an induction principle for paths in coequalizers. This is extremely useful, since we want to define functions out of spans in HITs. (← rework this sentence)

Summarizing this will be very technical, and may become its own chapter if I successfully formalize the proof in cubical agda. Otherwise it goes here.

Chapter 2

Homotopy Type Theory

It's cool. [13]

The purpose of this section is to give the enough prerequisites to follow the ensuing development [pretentious af]. It is not a complete introduction to Homotopy Type Theory. For a good introduction see Egbert Rijke's master thesis [10] and lecture notes [11], for a complete textbook see The Book [13].

2.0.1 Types and Propositions (and spaces?)

1. types represent propositions (and spaces)
2. implication and simple and/or ($\to, \times, +$)
3. quantifiers and dependent types (fibers) (Σ, Π)

2.0.2 Programs and Proofs (and terms?)

1. if types are propositions, how do we prove them?
2. *terms* of a type are *proofs* of a proposition

2.0.3 Identity Types

1. what about things that are equal?
2. J-rule (intuition: reflexive closure? groupoid structure?)
3. paths in space

2.0.4 Higher Inductive Types

1. inductive types: base case(s) and point generator(s)
2. example: $A + B, \mathbb{N}$
3. HIGHER inductive types: terms and identities
4. ie. points and paths between points (and paths between paths (and paths between paths between paths))
5. elimination rules? they need to go somewhere, but this might not be it

2.0.5 Cubical?

Why not take "= is a path" seriously?

Chapter 3

Version Control Systems

They're not always cool. [9]

Version control systems are ubiquitous in software development, where they help facilitate cooperation and documentation of the development process. Their basic use is to record (*commit*) changes to a codebase (*repository*). Systems may also include ways for the codebase to diverge (*branch*) into different versions, and ways to reunite (*merge*) these versions.

The purpose of this section is to introduce the terminology, requirements and hopes for models of version control systems,

What do we need?

1. terms
    - repository
    - patch
    - merge
    - branch?
2. requirements
    - repo - accurately represents contents

- patch - applicable in a context, groupoid structure
- merge - “pushout property”/reconcile, symmetric (for distributed systems), (do we need associativity as well?)

3. hopes/goals

   - repos - modular/composable, somehow polymorphic
   - patches - *semantic* in some sense
   - merge - easily definable [sic.], considers semantics of patches

Chapter 4

Conclusion

We did some things and they worked out — or maybe they didn't.

Bibliography

[1] 2021. URL: https://pijul.org/.

[2] 2021. URL: http://darcs.net/.

[3] Carlo Angiuli et al. "Homotopical patch theory". In: *Journal of Functional Programming* 26 (2016). ISSN: 14697653. DOI: 10.1017/S0956796816000198.

[4] Robin Houston. *Revisiting "On Editing Text"*. 2014. URL: https://bosker.wordpress.com/2014/06/19/revisiting-on-editing-text/.

[5] Dagit Jason. *Type-correct changes — a safe approach to version control implementation*. 2009. URL: https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/47429f27z.

[6] Nicolai Kraus and Jakob von Raumer. "Path Spaces of Higher Inductive Types in Homotopy Type Theory". In: (2019). arXiv: 1901.06022 [math.LO].

[7] Ian Lynagh. "An algebra of patches". In: (2006). URL: http://urchin.earth.li/~ian/conflictors/paper-2006-10-30.pdf.

[8] Samuel Mimram and Cinzia Di Giusto. "A Categorical Theory of Patches". In: *CoRR* abs/1311.3903 (2013). arXiv: 1311.3903. URL: http://arxiv.org/abs/1311.3903.

[9] Russell O'Connor. *Git is Inconsistent.* http://r6.ca/blog/20110416T204742Z.html. Apr. 2011.

[10] Egbert Rijke. "Homotopy Type Theory". In: (2012).

[11] Egbert Rijke. *Introduction To Homotopy Type Theory.* Lecture Notes: https://hott.github.io/HoTT-2019/images/hott-intro-rijke.pdf. 2019.

[12] Ganesh Sittampalam. "Some properties of Darcs patch theory". In: (2005). URL: http://urchin.earth.li/darcs/ganesh/darcs-patch-theory/theory/formal.pdf.

[13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: `https://homotopytypetheory.org/book`, 2013.

Appendix A

This is an appendix, if need be