Proposal of team software project

Department of Software Engineering

Faculty of Mathematics and Physics, Charles University

Solvers: Bc. Ondřej Krsička

Study program: Computer Science - Software and Data Engineering

Project title: Using CRDTs to enable collaborative editing in Denicek

Project type: Research project

Supervisor: Mgr. Tomáš Petříček, Ph.D.

Expected start: 1.11.2025

Expected end: 1.7.2026

1 Introduction

Denicek [1] is a document-based end-user programming substrate that can be used as the basis for the implementation of different programming systems. One such programming system is the web-based system *Webnicek*. Internally, *Denicek* relies on synchronization between document versions and it currently uses Operational Transformation, which is error prone and complex and requires a central server for synchronization. This project will create *MyDenicek*, a backend built on top of CRDTs. This will make it easier to implement programming systems that follow the principles of local-first software. *MyWebnicek* will be a new web-based programming system built on top of the new substrate, providing functionality similar to *Webnicek*.

2 Denicek end-user programming experiences

Denicek: Computational Substrate for Document-Oriented End-User Programming [1] provides the following end-user programming experiences:

- Collaborative editing
- Programming by demonstration (users can record actions and program a button to replay them on click)
- Incremental recomputation (formulas depending on changed values are invalidated and automatically recomputed)
- Schema change control (when a value is wrapped or unwrapped in an element, all references to it are updated)
- End-user debugging (values on which a formula depends can be highlighted for better understanding of the result)
- Concrete programming (copying and pasting formulas between contexts while preserving dependencies; when the original changes, the copied formula updates accordingly)

3 Denicek usage

Two end-user programming environments are currently built on top of the *Denicek* substrate:

- *Webnicek*: for creating interactive HTML documents
- *Datnicek*: for data science use cases similar to Jupyter Notebooks

The use cases of Denicek are illustrated by the following formative examples:

- **Counter app:** The user creates a document with value 1, wraps it in a formula (1+1), records the process, and replays it on button click: producing a simple counter.
- **Todo app:** Similar to the counter app, but the recorded actions add text from an input field as a new list item.
- **Conference list:** Alice refactors a list of speakers from comma-separated items (name, email) into a table with columns (name, email). Meanwhile, Bob adds a new list item. The merged result is a table containing all speakers, including Bob's addition.
- **Conference budget:** Building on the conference list, formulas depending on its values are automatically recomputed during refactoring.
- **Hello world:** The user makes the first line of a list of sentences lowercase, then capitalizes the first letter, copies the formula, and applies it to the whole list.
- **Traffic accidents:** Omitted from this research project, as it relates more closely to *Datnicek*.

4 Limitations of Denicek and goals of this research project

The current *Denicek* synchronization layer is implemented using Operational Transformation (OT). However, OT is complex, error-prone, and requires a central synchronization server, which does not satisfy the requirements of local-first software [6].

The goal of this research project is to create an alternative to *Webnicek* built on top of a CRDT-based local-first substrate. The new system should support the same end-user programming experiences demonstrated by the formative examples, possibly using a revised set of primitive operations. It should also maintain a clear separation between the user interface and the CRDT substrate to enable future development of a *Datnicek* alternative based on the same backend.

The resulting systems will be called *MyDenicek* and *MyWebnicek*, inspired by *MyWebstrates* [7]: a local-first, CRDT-based alternative to Webstrates [8]. Both parts will be implemented in TypeScript.

The primary deliverable of the project will be the CRDT-based backend, *MyDenicek*, implemented as a reusable library. *MyWebnicek* will be developed as a prototype/demo application built on top of *MyDenicek*; it is intended as an illustrative demo rather than a fully production-ready web application.

4.1 MyWebnicek functional requirements

MyWebnicek will have the following functionalities:

- Renderer of the final document
- Navigation through the document
- Commandline for user to perform primitive actions
- History view of user edit actions
- Sharing mechanism for collaborative editing via network
- Conflict resolution interface

4.2 MyDenicek functional requirements

The core library (*MyDenicek*) will satisfy the following requirements:

- Provide a way of representing HTML-like structured documents similar to those in Denicek
- Expose an API for modifying the underlying document through edit actions similar to those in Denicek
- Provide a CRDT-based operation for merging divergent versions of the document
- Support a mechanism for detecting and resolving merge conflicts akin to Grove

5 Schedule

The project is expected to run for 8 months. The expected schedule is:

- Task 1 (Studying of materials, analysis and prototyping) - Month 1-3
- Task 2 (Production-ready MyDenicek and MyWebnicek implementation) - Month 4-7
- Task 3 (Textual and Video Documentation) - Month 8
- Task 4 (Testing and evaluation) - Month 8

6 Team structure

The work will be coordinated by the supervisor. The student will be responsible for the technical aspects of the project (developing the Grove encoding, prototype implementation) and evaluation. The design of the system will be developed in collaboration between the supervisor, the student and Jonathan Edwards (Denicek co-author).

7 Related work

This work directly builds on the Denicek system presented at *ACM UIST 2025* [1]. It extends ongoing research on Conflict-free Replicated Data Types (CRDTs), as summarized in Preguiça's overview [2], and is informed by practical open-source CRDT frameworks (Yjs, Automerge). The project will first investigate the applicability of Grove's typed, patch-based model [3] to Denicek's document structures and editing workflows, while keeping the option to adopt or hybridize with other CRDTs if Grove proves unsuitable for some features.

References

[1] T. Petříček, et al. *Denicek: Computational Substrate for Document-Oriented End-User Programming.* In Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25). no. 32, pp. 1–19.

[2] N. Preguiça. *Conflict-free Replicated Data Types: An Overview.* 2018. `https://arxiv.org/abs/1806.10254`.

[3] M. D. Adams, et al. *Grove: A Bidirectionally Typed Collaborative Structure Editor Calculus.* ACM, 2024. DOI: `https://doi.org/10.1145/3704909`.

[4] A. Cypher, et al. (eds.) Watch what I do: programming by demonstration. MIT Press, 1993.

[5] J. Edwards, et al. Schema Evolution in Interactive Programming Systems. The Art, Science, and Engineering of Programming, 9(1), 2-1. 2024.

[6] M. Kleppmann, et al. Local-first software: you own your data, in spite of the cloud. Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. 2019.

[7] Clemens Nylandsted Klokmose, James R. Eagan, and Peter van Hardenberg. 2024. MyWebstrates: Webstrates as Local-first Software. In Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 42, 1–12. https://doi.org/10.1145/3654777.3676445

[8] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15). Association for Computing Machinery, New York, NY, USA, 280–290. https://doi.org/10.1145/2807442.2807446

[9] Loro. *Loro CRDT Library.* `https://loro.dev/`

A MyDenicek document structure

A *MyDenicek document* is represented via a `DocumentView` class that encapsulates the tree structure, using **Loro CRDTs** [9] for synchronization:

```
class DocumentView {
  getRootId(): string | null;
  getNode(id: string): NodeData | null;
  getChildIds(parentId: string): string[];
  getParentId(nodeId: string): string | null;
  getAllNodeIds(): string[];
  *walkDepthFirst(): Generator<{ node: NodeData; depth; parentId }>;
```

```
}

// Node data types (no children array - use getChildIds instead)
interface ElementNodeData {
  id: string;
  kind: "element";
  tag: string;
  attrs: Record<string, unknown>;
}

interface ValueNodeData {
  id: string;
  kind: "value";
  value: string;
}

type NodeData = ElementNodeData | ValueNodeData;
```

Internally, the document is stored as a `LoroTree`—Loro's native movable tree CRDT that handles concurrent structural edits, move operations, and conflict resolution. The `DocumentView` class is the *public API* that:

- **Hides CRDT internals**: No Loro types are exposed; applications work with plain TypeScript objects
- **Enables O(1) lookup**: Internal index maps allow efficient node, parent, and children lookups
- **Prevents direct mutation**: Users access data through methods instead of property access
- **Simplifies rendering**: React components receive a stable view for efficient diffing

B MyDenicek operations

Primitive operations supported by the CRDT-based system:

- Add child node (element or value) to a parent
- Add sibling node (before or after reference node)
- Delete node
- Update node tag
- Update node attribute
- Edit value content (character-level text editing)
- Wrap node in new parent element

Conflict resolution uses deterministic wrapper IDs (`w-${nodeId}`) for wrap operations and Last-Writer-Wins (LWW) semantics for concurrent tag/value edits.