



CRDTs for Spreadsheets:

Are they useful?

Nicolás Hatcher

Local-First Conf 2026

Berlin

IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

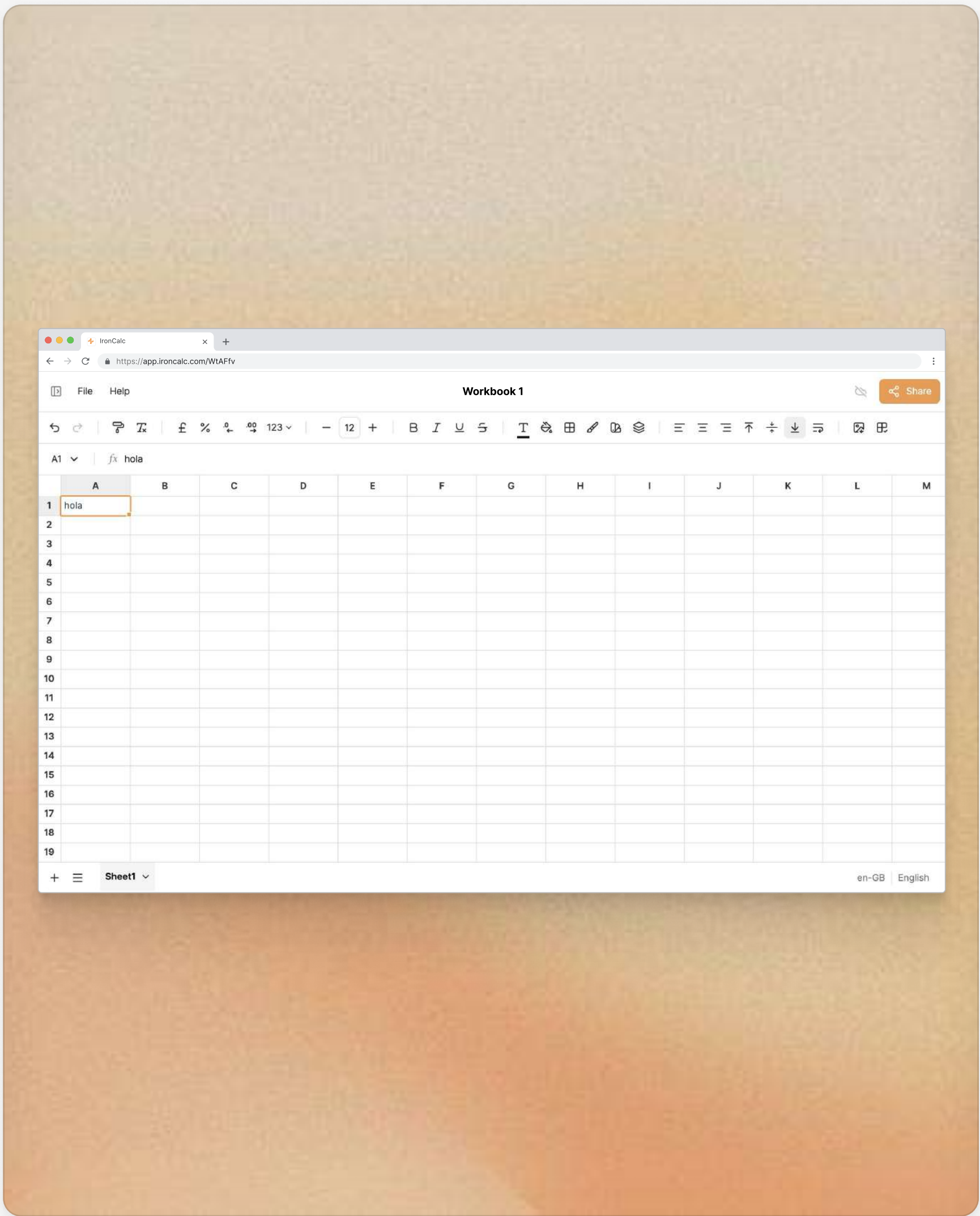
Supported by



Horizon Europe programme
via ELFA Consortium



The nInet Foundation
via NG10 Core Fund



IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

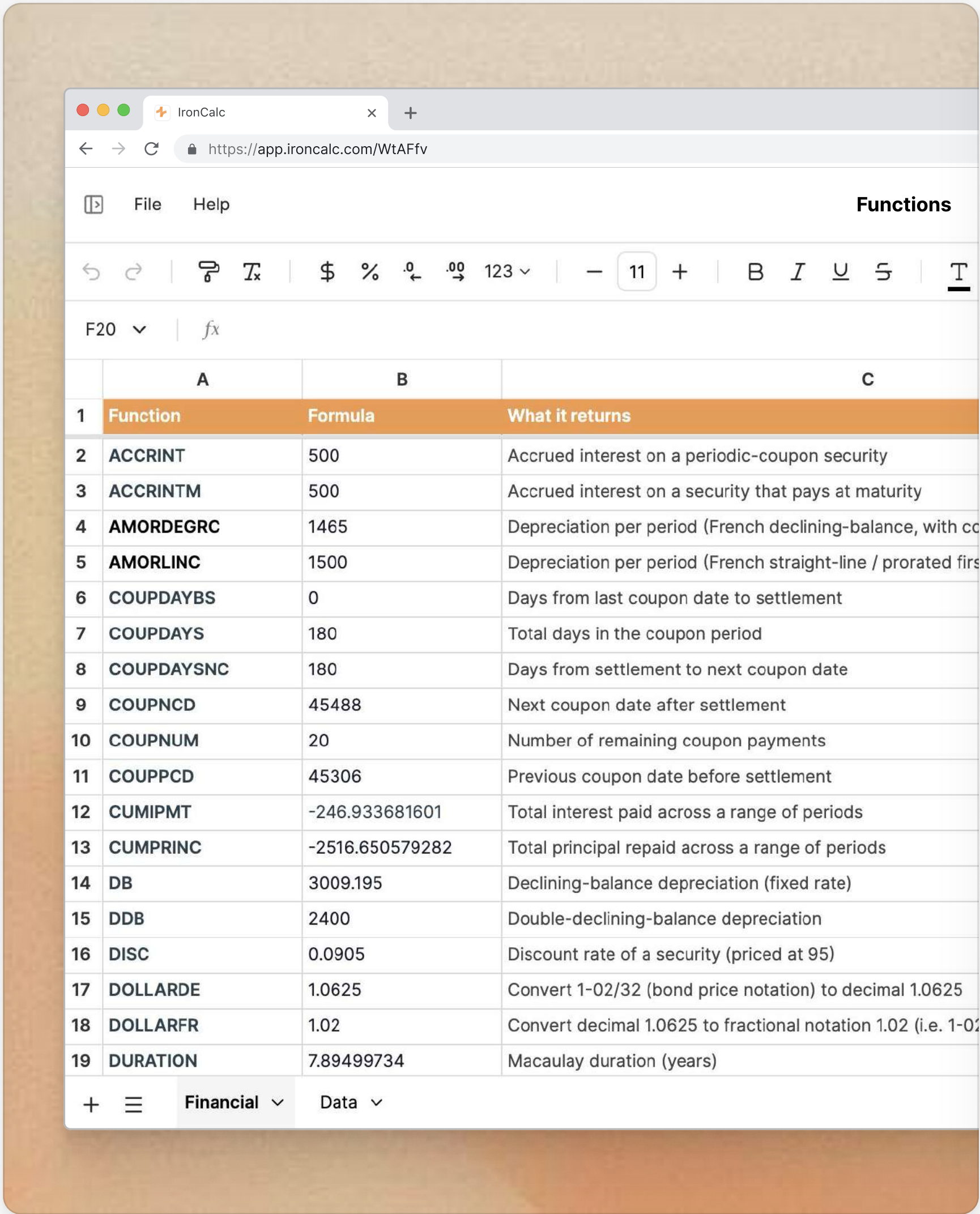
Supported by



Horizon Europe programme
via ELFA Consortium



The nInet Foundation
via NG10 Core Fund



IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

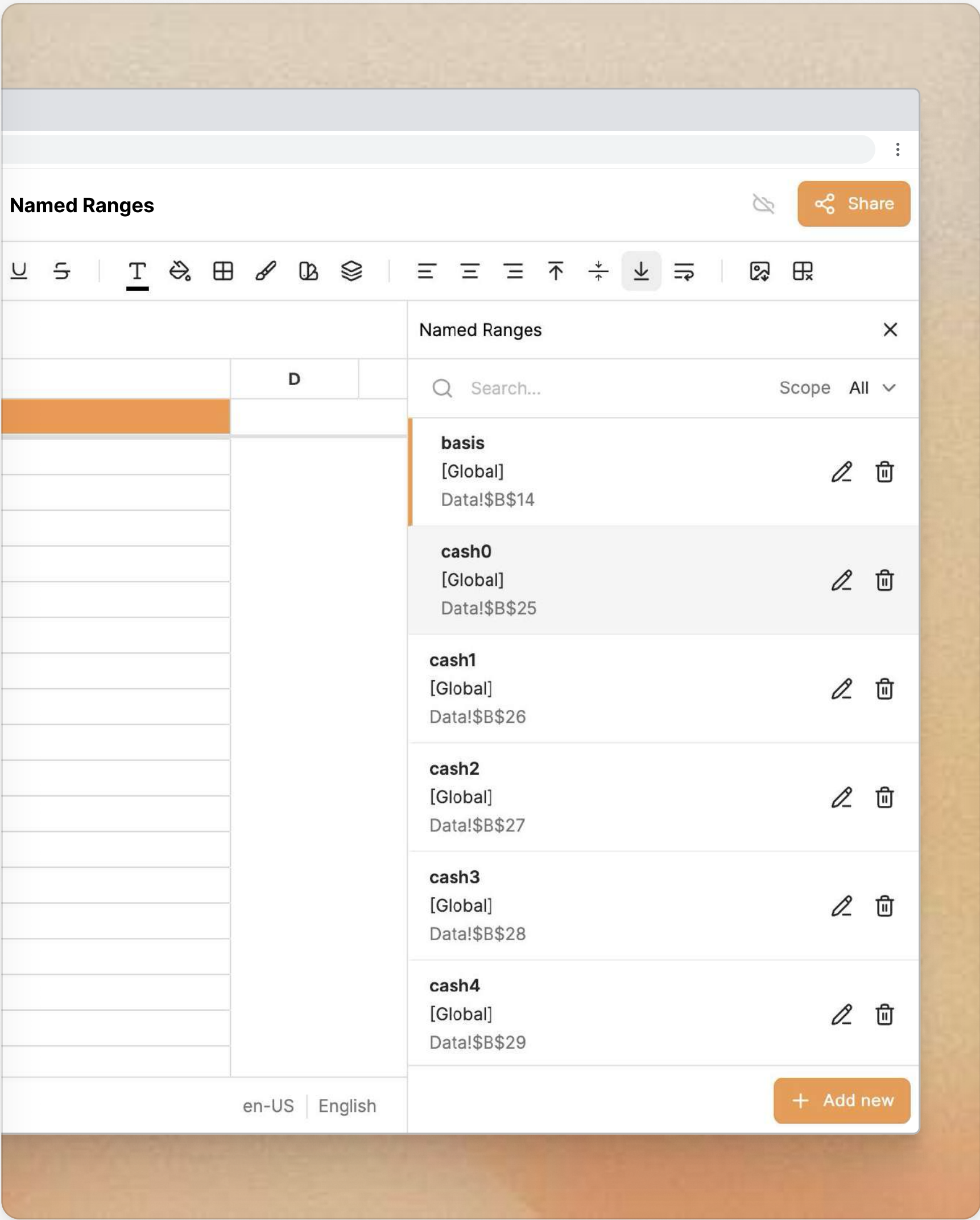
Supported by



Horizon Europe programme
via ELFA Consortium



The nInet Foundation
via NG10 Core Fund



IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

Supported by



Horizon Europe programme
via ELFA Consortium



The nInet Foundation
via NG10 Core Fund

IronCalc

https://app.ironcalc.com/WtAFfv

FileHelp

↶↷

🔍

ℳ%

123

13

B6Berlin

	A	B	C	D	E
1					
2		City	Country	Population	
3		Baku	Azerbaijan	2,351,300	
4		Barcelona	Spain	1,731,649	
5		Belgrade	Serbia	1,682,720	
6		Berlin	Germany	3,685,265	
7		Birmingham	United Kingdom	1,183,618	
8		Bucharest	Romania	1,709,458	
9		Budapest	Hungary	1,685,209	
10		Cologne	Germany	1,024,210	
11		Hamburg	Germany	1,867,832	
12		Istanbul	Turkey	15,754,053	
13		Kazan	Russia	1,329,825	
14		Kharkiv	Ukraine	1,421,225	
15		Krasnodar	Russia	1,154,885	

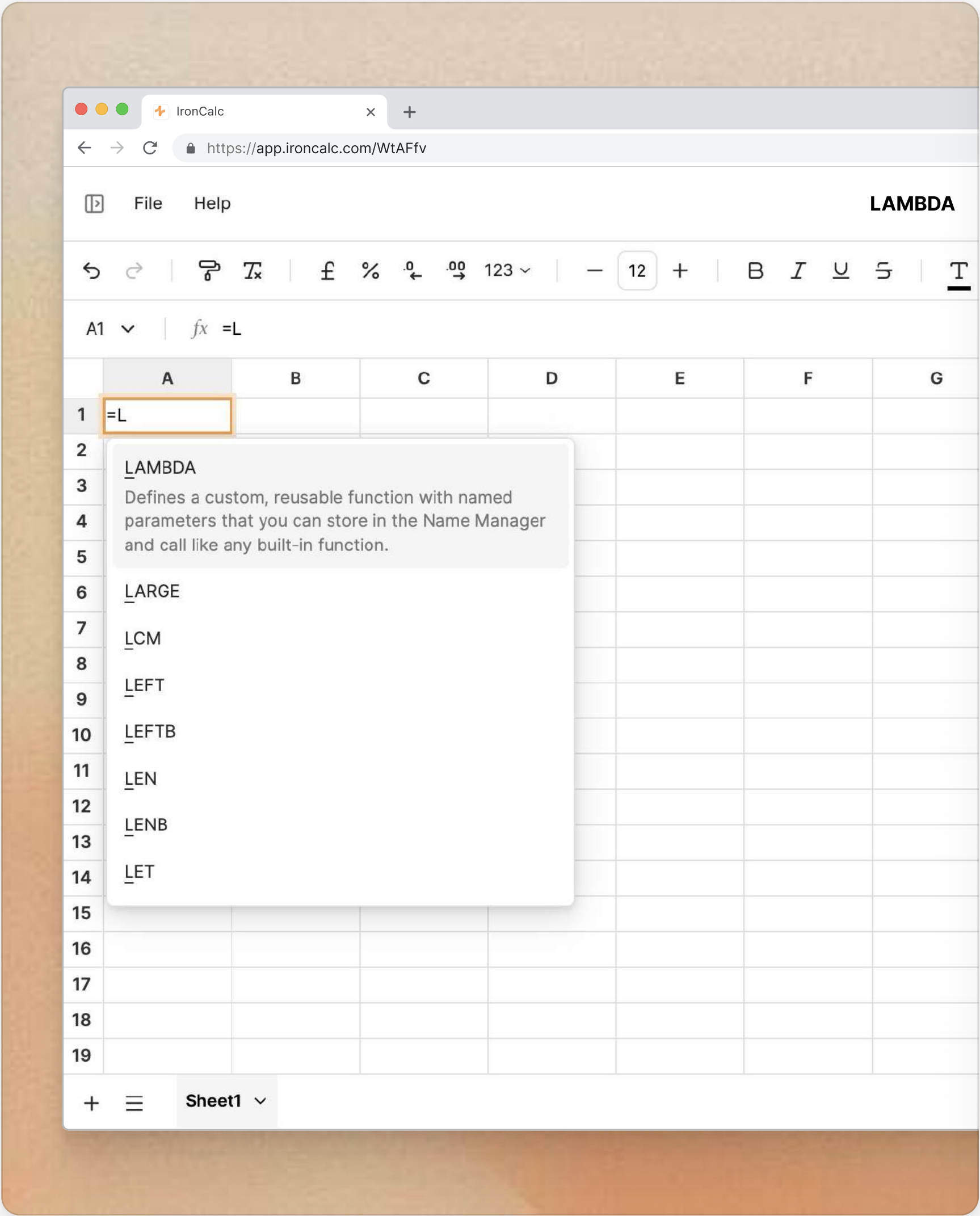
IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

Supported by

 Horizon Europe programme
via ELFA Consortium

 The nInet Foundation
via NG10 Core Fund



IronCalc is a full fledged spreadsheet engine.

We have almost 500 functions, defined names, conditional formatting, Lambda functions, dynamic arrays, named cell styles, i18n, you name it!

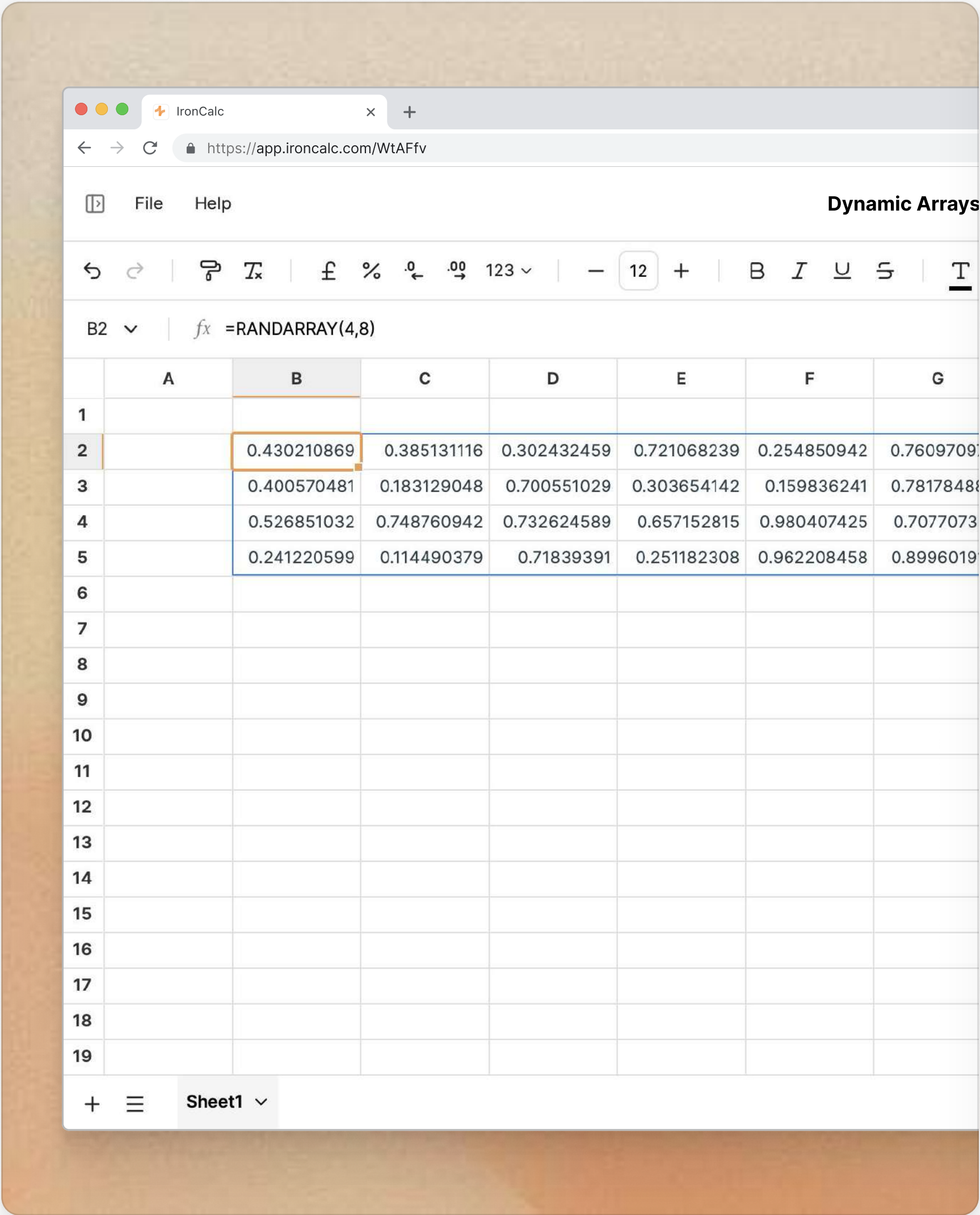
Supported by



Horizon Europe programme
via ELFA Consortium



The nlNet Foundation
via NG10 Core Fund



But it is not collaborative

But it is not collaborative yet

Collaborative spreadsheets: **What do we want?**

We want to deliver best quality software.

It is real software, not a theoretical spreadsheet, not a toy model.

-  Spreadsheet that can be used collaboratively
-  Eventual consistency
-  Preserve intent as much as possible
-  Don't make the user think/decide (git merge conflicts!)
-  Do we need a central authoritative server?

Scenario 1

Real time collaborative editing.

There is a central server. Although it is real time we have all the problems of the internet, packages drop, order issues, network reliability,...



Scenario 2

Offline use.

In a plane, remote places or because the user wants to focus.



**We might use
the **same** approach in both cases
or **different** approaches.**

We might decide NOT to solve #2, for instance. In case #1 quick broadcast and application of everyone else's changes might solve 90% of the problems.

Common problems
(to any algorithm)

#1

Edit cells

Changes things like row height, style, strings are indexed.

Affected	Changed	Affected

Changed Affected

#2

Clear cells

Clear only content and also style

#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	Changed	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!

Changed Affected

#3

Style cells

Issues with borders, practical issues with indices.

Changed Affected

#4

Rows and Columns

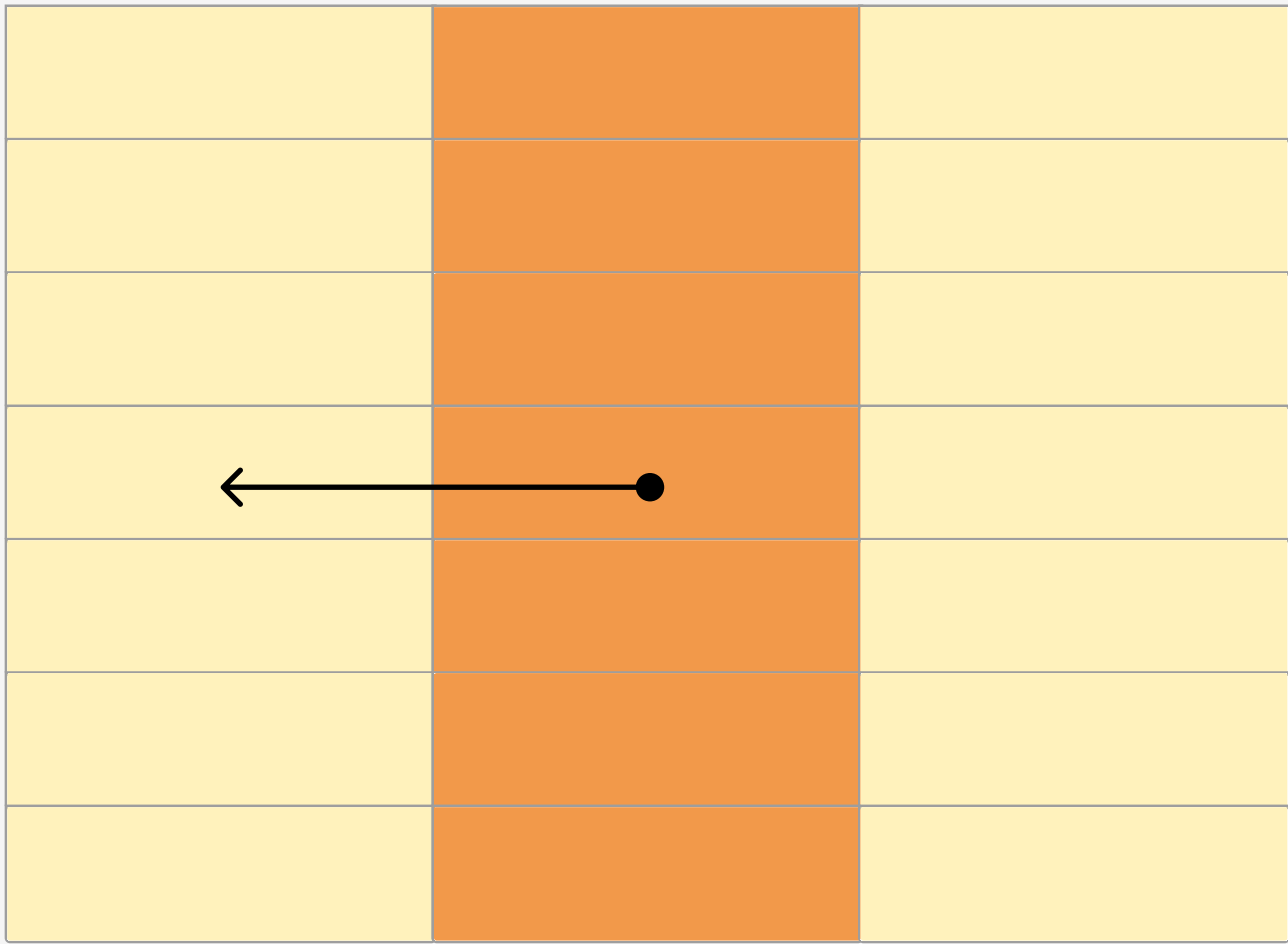
Insert and delete.

Changed Affected

#5

Rows and Columns

Move.

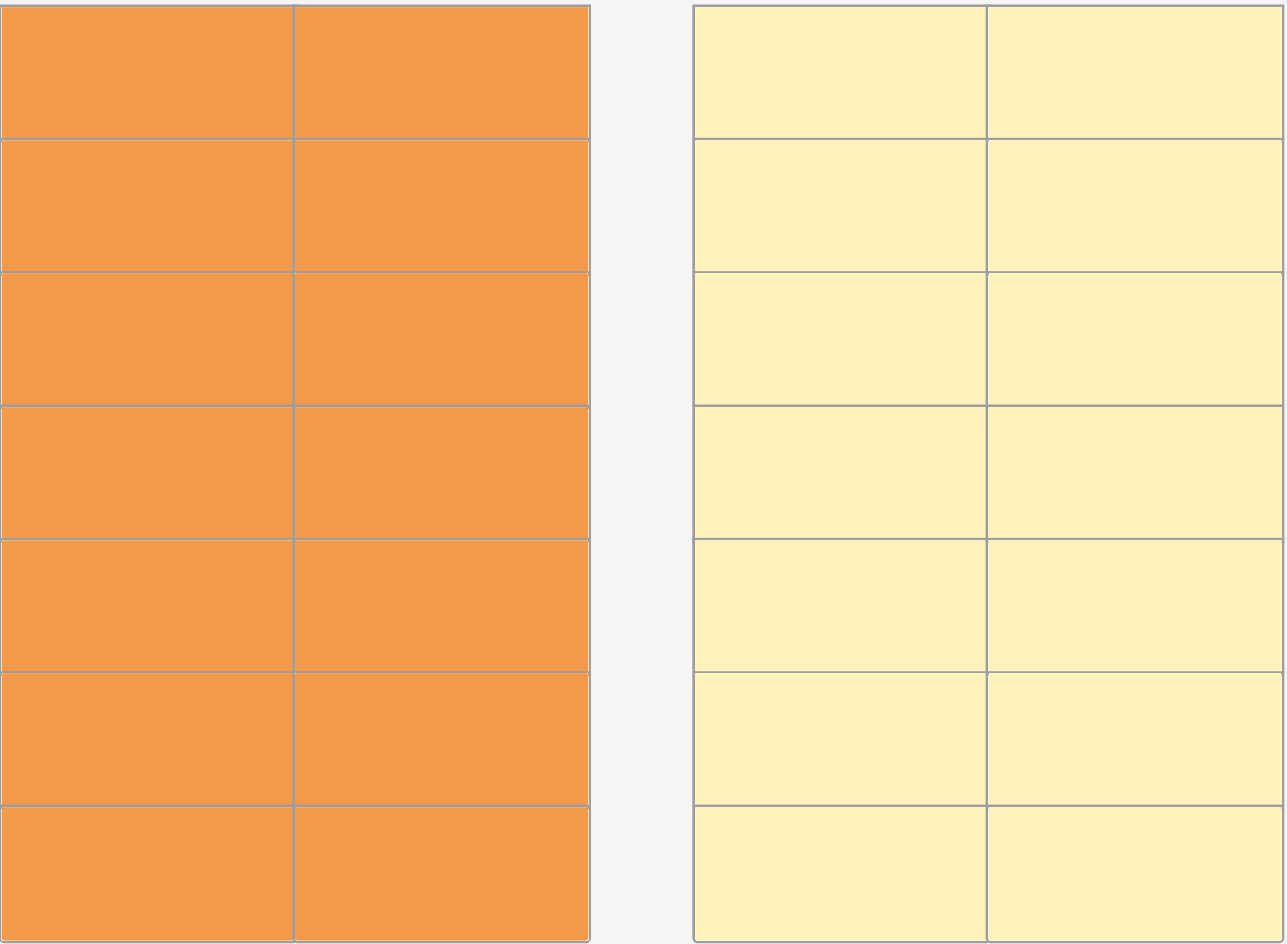


● Changed ● Affected

#6

Sheets

Delete and duplicate.



● Changed ● Affected

#7

Named Ranges

Create, update and delete.

	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!
#REF!	#REF!	#REF!

● Changed ● Affected

#8

Complex computations

Like spill formulas.

	=SEQUENCE(10)	

● Changed ● Affected

Some examples

AegisSheet: Pfeil, Scandurra, Hass (2026)

Google Sheets

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Last-write-wins (LWW)	–	Remove wins	Move and edit
InsertR/C	–	Two distinct	Insert and remove	Insert at pre-move position
RemoveR/C	Remove wins	Insert and remove	Remove	Move wins
MoveR/C	Move wins (discard edit)	Insert at pre-move position	Move wins	Last move wins, earlier move creates empty row(column)

Google Sheets w/ Undo

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Revert both	–	Revert both	Revert both
InsertR/C	–	Revert insert	Revert remove	Revert move, but to wrong position
RemoveR/C	No-op (cols at old indices are highlighted)	Revert insert	Bring back removed cells	Revert both
MoveR/C	No-op (highlight unchanged rows)	Revert insert	Duplicate row (column)	Clear row (column)

Can we do better?
Are CRDTs useful?

They are...

They are... But implementation is difficult

Edit cells
Changes things like row height, style, strings are indexed.

Clear cells
Clear only content and also style

Style cells
Issues with borders, practical issues with indices.

Rows and Columns
Insert and delete.

Rows and Columns
Move.

Sheets
Delete and duplicate.

Named Ranges
Create, update and delete.

Complex computations
Like spill formulas.

CRDTs intuition:

Can we make all those user operations commutative?

CRDTs in spreadsheets are pretty much a research project:

A Study of Semantics for CRDT-based Collaborative Spreadsheets

Elena Yanakieva
yanakieva@cs.uni-kl.de
University of Kaiserslautern-Landau
Kaiserslautern, Germany

Philipp Bird
p_bird14@cs.uni-kl.de
University of Kaiserslautern-Landau
Kaiserslautern, Germany

Annette Bieniusa
bieniusa@cs.uni-kl.de
University of Kaiserslautern-Landau
Kaiserslautern, Germany

Abstract
Online collaboration is becoming prevalent in our day-to-day work. As commercial applications show, next to texts, spreadsheets are an essential tool for storing and organizing shared data. However, concurrent modifications of a collaborative spreadsheet can lead to unexpected results when they reflect implementation decisions rather than user intention. With this paper, we systematically discuss spreadsheet operations and their semantics and propose intention-preserving designs in a concurrent decentralized setting, thus supporting offline operations. We further explore different data models for shared spreadsheets based on composed Conflict-free Replicated Data Types (CRDTs) and give an implementation in the local-first framework Yjs.

CCS Concepts: • Applied computing → Text editing; • Theory of computation → Theory and algorithms for application domains.

Keywords: offline-first, collaborative spreadsheet, CRDTs

ACM Reference Format:
Elena Yanakieva, Philipp Bird, and Annette Bieniusa. 2023. A Study of Semantics for CRDT-based Collaborative Spreadsheets. In *10th Workshop on Principles and Practice of Consistency for Distributed Data (PPoC '23)*, May 8, 2023, Rome, Italy. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3578358.3591324>

1 Introduction
Collaborating with others is essential to our everyday lives, whether for work or private matters. In this context, shared data can take many forms: text, tables, and images are representative selections for such data. With the help of collaborative editors, users can concurrently create and modify

Semantics for CRDTs
Yanakieva, Bird, Bieniusa, 2023

AegisSheet: A Compositional CRDT for Collaborative Spreadsheets

Florian Pfeil^{*}
Technische Universität Darmstadt
Darmstadt, Hesse, Germany
florian.pfeil@stud.tu-darmstadt.de

David Scandurra^{*}
Technische Universität Darmstadt
Darmstadt, Hesse, Germany

Julian Haas
Technische Universität Darmstadt
Darmstadt, Hesse, Germany
haas@cs.tu-darmstadt.de

Abstract
Applications that support online collaboration over shared documents such as text files and spreadsheets are becoming increasingly popular. While collaborative text editing is well studied and multiple specialized data structures exist, there has been surprisingly little innovation in the field of collaborative spreadsheets, which allow multiple users to collaboratively organize data, and perform data-driven computations: Major providers often still resort to "last-writer-wins" semantics for common conflict scenarios, resulting in potential data loss that makes these applications ill-equipped for asynchronous and offline collaboration. In this paper, we study the state of the art in collaborative spreadsheets and propose a novel conflict-free replicated data type (CRDT) for spreadsheets derived from first principles. Our design goes beyond existing solutions by supporting move operations, ranged computations, undo, and persistent cell conflicts. Instead of relying on one monolithic algorithm, our implementation follows a compositional approach that derives the spreadsheet CRDT from a composition of novel and existing CRDTs. By building on existing data structures, this approach allows for simpler correctness reasoning while keeping the design flexible enough to be adapted to different use cases.

CCS Concepts: • Information systems → Asynchronous editors; • Theory of computation → Distributed algorithms; • Applied computing → Text editing.

Keywords: CRDT, local-first, ARDT, collaborative spreadsheet

^{*}Both authors contributed equally to this research.

Authors' Contact Information: Florian Pfeil, Technische Universität Darmstadt, Darmstadt, Hesse, Germany, florian.pfeil@stud.tu-darmstadt.de; David Scandurra, Technische Universität Darmstadt, Darmstadt, Hesse, Germany; Julian Haas, Technische Universität Darmstadt, Darmstadt, Hesse, Germany, haas@cs.tu-darmstadt.de.


This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).

AegisSheet
Pfeil, Scandurra, Hass, 2026

October 5, 2023

Conflict-free Replicated Spread Sheets

#crdt #javascript #distributed-systems

In this blog post we're going to cover a concepts and implementation behind collaborative 2-dimensional tables, with set of operation that could make them useful to work as spread sheets - popular in products like MS Excel or Google Sheets. There are many open source and commercial alternatives on the market, but at the moment of writing this post a few of them offer any sort of collaborative features, even fewer are capable to extend that to offline-first capabilities.

CRDT table architecture

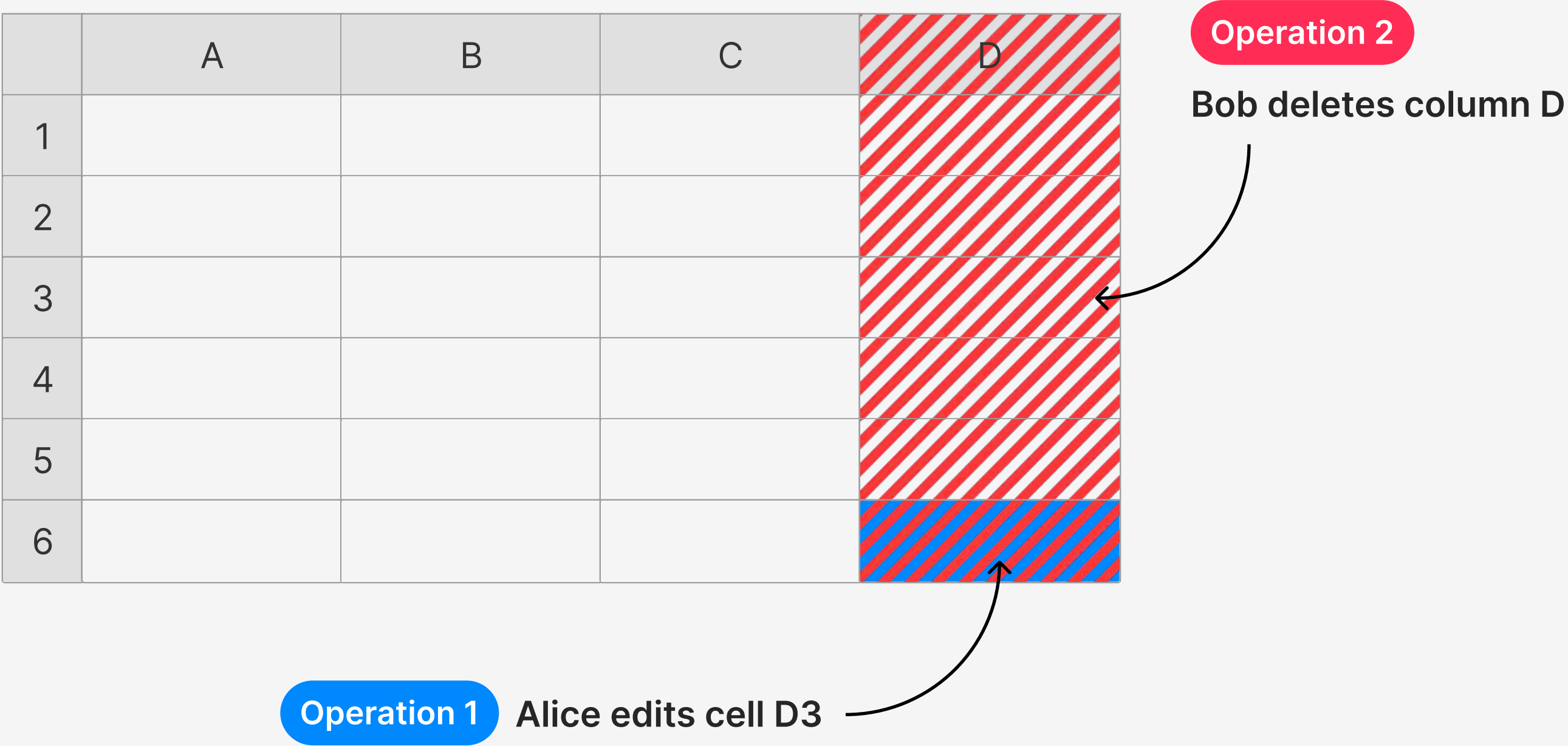
Below we'll be describing the technical overview of the data structure, which source code can be find in [this Github repository](#).

Before going to implementation details, let's specify the characteristics of the spread sheets workloads. Once data volume grows, its impossible to optimize the data structure for all work patterns. We need to pick our battles. For this reason we need to construct some initial assumptions:

- Some spread sheets can consists of hundreds of columns and (in some cases) hundreds of thousands of rows. For this reason, the size of metadata per row (and per cell) matters.
- Relative position of rows is usually not that important - unless we apply specific ordering rules (like order rows by column X), the insertion order rarely changes the

Conflict-free Replicated Spreadsheets
Bartosz Sypytkowski, 2023

How can those operations be commutative?



How can those operations be commutative?

Operation 2 × Operation 1 = Operation 1 × Operation 2 = Edit cell D3

Some examples

AegisSheet: Pfeil, Scandurra, Hass (2026)

Google Sheets

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Last-write-wins (LWW)	–	Remove wins	Move and edit
InsertR/C	–	Two distinct	Insert and remove	Insert at pre-move position
RemoveR/C	Remove wins	Insert and remove	Remove	Move wins
MoveR/C	Move wins (discard edit)	Insert at pre-move position	Move wins	Last move wins, earlier move creates empty row(column)

Google Sheets w/ Undo

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Revert both	–	Revert both	Revert both
InsertR/C	–	Revert insert	Revert remove	Revert move, but to wrong position
RemoveR/C	No-op (cols at old indices are highlighted)	Revert insert	Bring back removed cells	Revert both
MoveR/C	No-op (highlight unchanged rows)	Revert insert	Duplicate row (column)	Clear row (column)

Some examples

AegisSheet: Pfeil, Scandurra, Hass (2026)

Ours

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Create conflict	–	Remove wins	Move and edit (at new position)
InsertR/C	–	Create both	Insert and remove	Insert (at old position) and move
RemoveR/C	Discard remove; apply edit	Insert and remove	Remove	Move wins
MoveR/C	Move and edit (at new position)	Insert (at old position) and move	Move wins	Last move wins

Ours w/ Undo

Operation	EditCell	InsertR/C	RemoveR/C	MoveR/C
EditCell	Create conflict (between initial value and edit 2)	–	No-op (remove already canceled by edit)	Revert move (retain edit)
InsertR/C	–	Revert local insert	Revert remove	Revert move (do not alter position of insert)
RemoveR/C	Revert edit	Revert insert	Revert both	Revert both
MoveR/C	Revert edit	Revert insert	No-op (remove already canceled by move)	Revert both

That's all!

(For now)

Reach out

nicolas@ironcalc.com



IronCalc