Universität St.Gallen

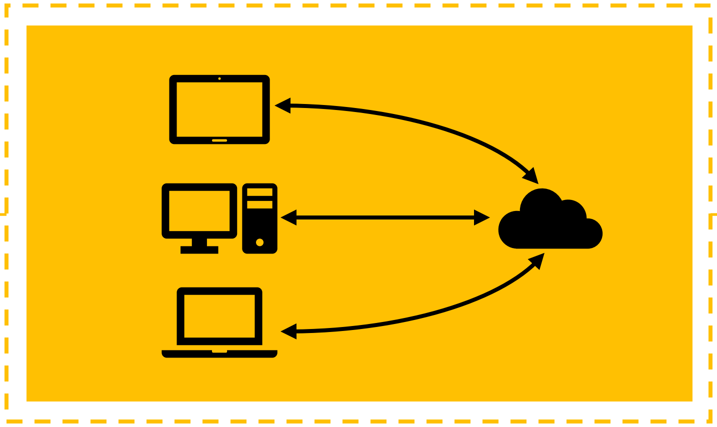
Bachelor Project

Helix

Local First Software

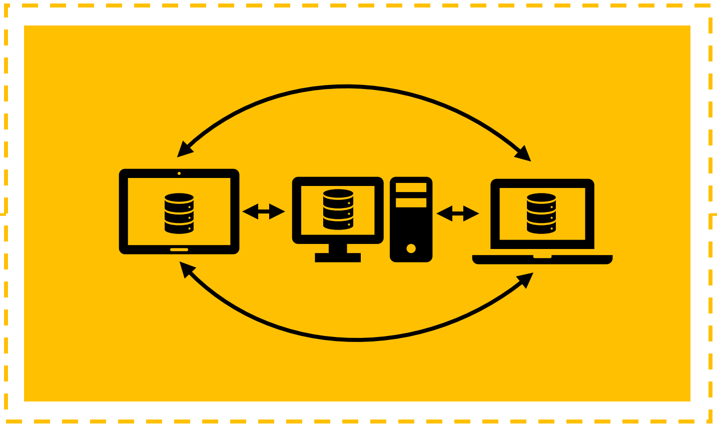
Teo Field-Marsham
Max Beringer

What is Local First



Cloud First

- Centralized Data Storage
- Requires Internet Access
- Cost for high Usage



Local First

- Offline availability
- Full Data Ownership
- Data privacy

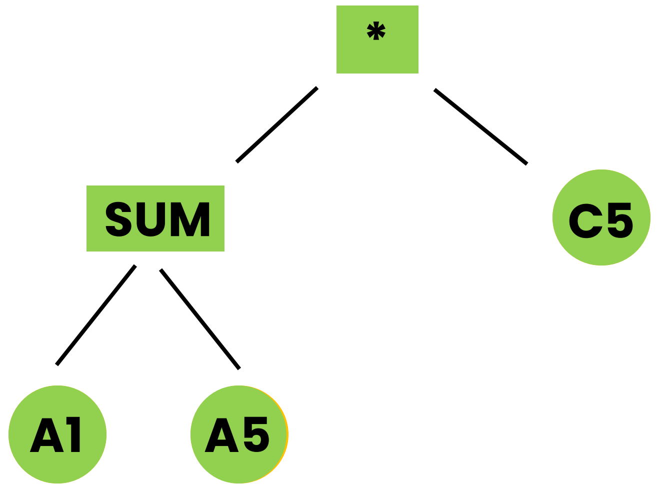
Helix

- Collaborative Excel Formula Editing
- Automatic Synchronization
  - Conflict-free
- Merge Algorithm
  - Idempotent
  - Associative
  - Communicative

Helix Magic under the Hood

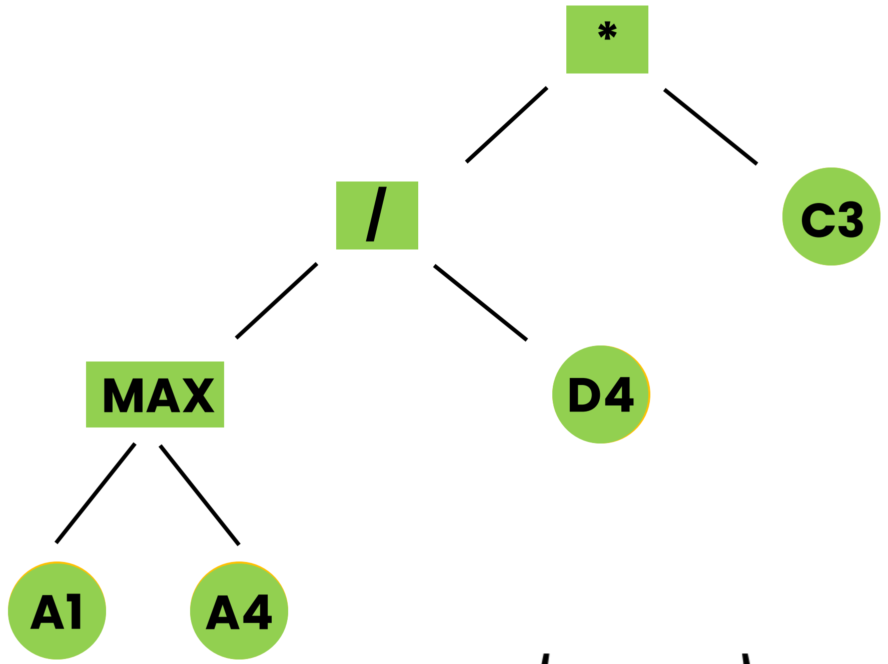
SUM(A1 ; A5) * C5

=



(MAX (A1 ; A4) / D4) * C3

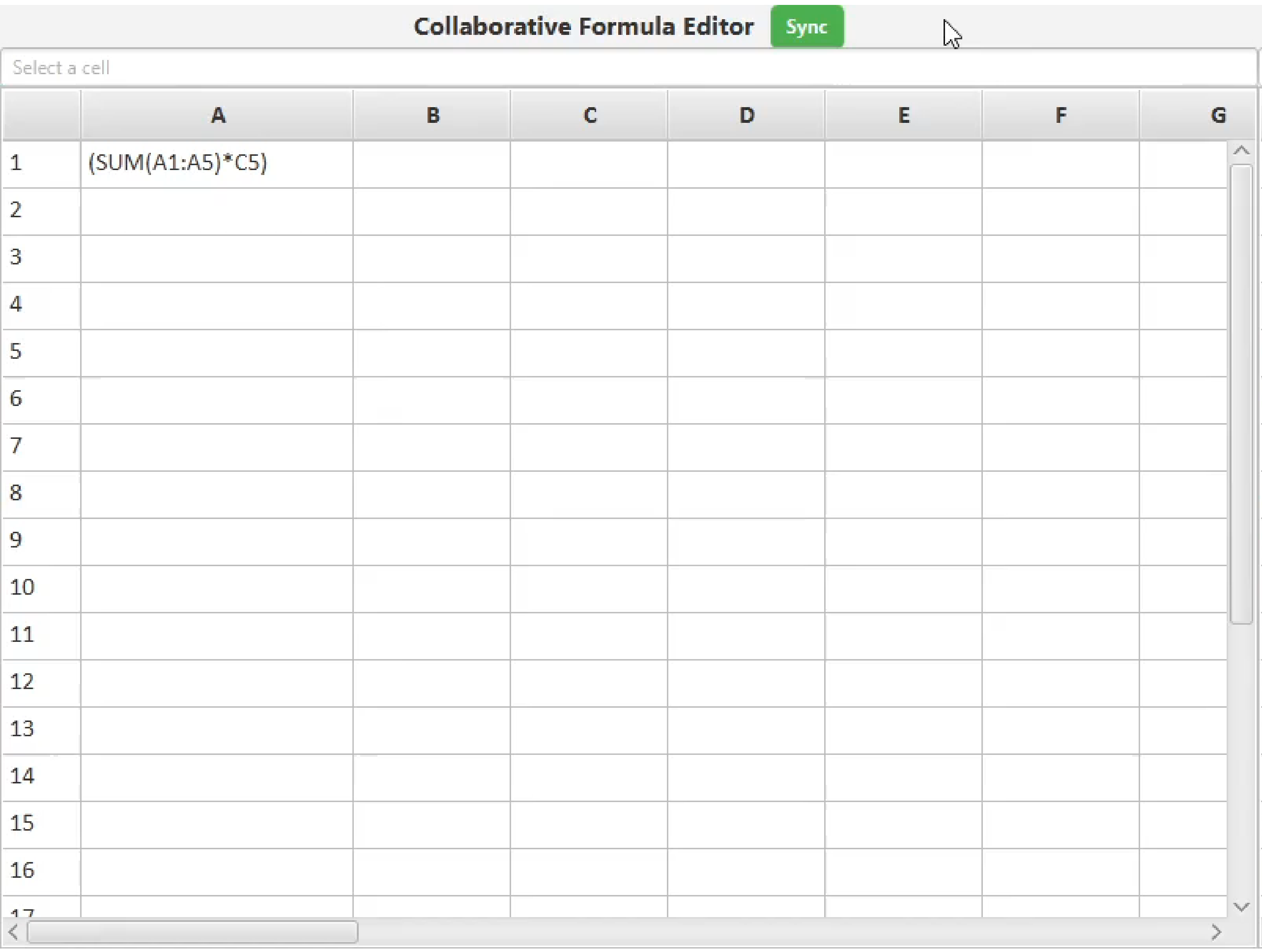
=



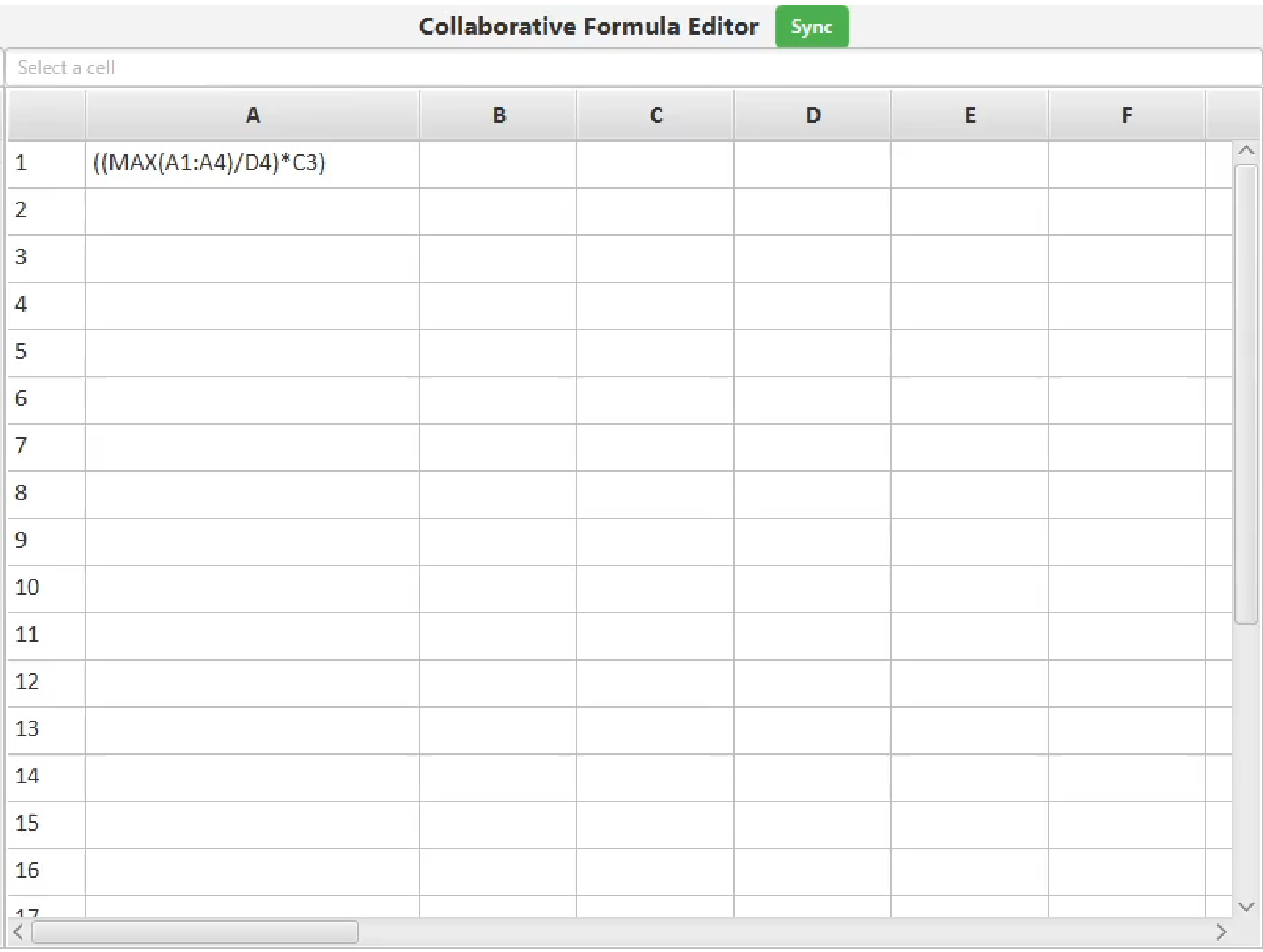
Alice: $\begin{pmatrix} \text{Alice:} & 3 \\ \text{Bob:} & 2 \\ \text{Claire:} & 1 \end{pmatrix}$ = (MAX (A1 ; A5) / D4) * C5

Bob: $\begin{pmatrix} \text{Alice:} & 2 \\ \text{Bob:} & 3 \\ \text{Claire:} & 1 \end{pmatrix}$

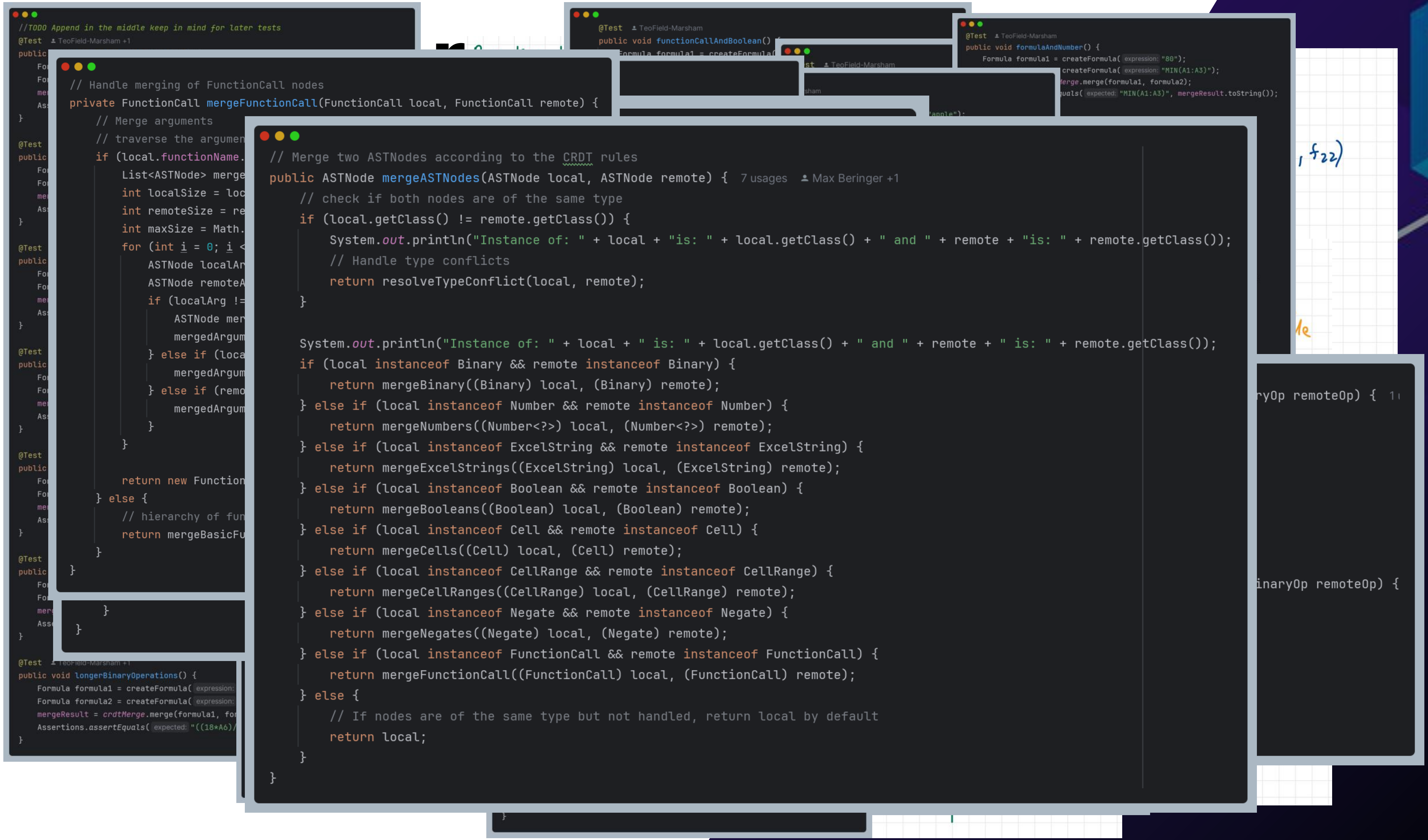
Inside the Software



	A	B	C	D	E	F	G
1	(SUM(A1:A5)*C5)						
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							



	A	B	C	D	E	F
1	((MAX(A1:A4)/D4)*C3)					
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						



```java
//TODO Append in the middle keep in mind for later tests
@Test  TeoField-Marsham +1
public
```

```java
// Handle merging of FunctionCall nodes
private FunctionCall mergeFunctionCall(FunctionCall local, FunctionCall remote) {
    // Merge arguments
    // traverse the argumen
    if (local.functionName.
        List<ASTNode> merge
        int localSize = loc
        int remoteSize = re
        int maxSize = Math.
        for (int i = 0; i <
            ASTNode localAr
            ASTNode remoteA
            if (localArg !=
                ASTNode mer
                mergedArgum
            } else if (loca
                mergedArgum
            } else if (remo
                mergedArgum
            }
        }

        return new Function
    } else {
        // hierarchy of fun
        return mergeBasicFu
    }
}
```

```java
@Test  TeoField-Marsham
public void functionCallAndBoolean() {
```

```java
@Test  TeoField-Marsham
public void formulaAndNumber() {
    Formula formula1 = createFormula( expression: "80");
                       createFormula( expression: "MIN(A1:A3)");
                 Merge.merge(formula1, formula2);
              uals( expected: "MIN(A1:A3)", mergeResult.toString());
```

```java
// Merge two ASTNodes according to the CRDT rules
public ASTNode mergeASTNodes(ASTNode local, ASTNode remote) {  7 usages   Max Beringer +1
    // check if both nodes are of the same type
    if (local.getClass() != remote.getClass()) {
        System.out.println("Instance of: " + local + "is: " + local.getClass() + " and " + remote + "is: " + remote.getClass());
        // Handle type conflicts
        return resolveTypeConflict(local, remote);
    }

    System.out.println("Instance of: " + local + " is: " + local.getClass() + " and " + remote + " is: " + remote.getClass());
    if (local instanceof Binary && remote instanceof Binary) {
        return mergeBinary((Binary) local, (Binary) remote);
    } else if (local instanceof Number && remote instanceof Number) {
        return mergeNumbers((Number<?>) local, (Number<?>) remote);
    } else if (local instanceof ExcelString && remote instanceof ExcelString) {
        return mergeExcelStrings((ExcelString) local, (ExcelString) remote);
    } else if (local instanceof Boolean && remote instanceof Boolean) {
        return mergeBooleans((Boolean) local, (Boolean) remote);
    } else if (local instanceof Cell && remote instanceof Cell) {
        return mergeCells((Cell) local, (Cell) remote);
    } else if (local instanceof CellRange && remote instanceof CellRange) {
        return mergeCellRanges((CellRange) local, (CellRange) remote);
    } else if (local instanceof Negate && remote instanceof Negate) {
        return mergeNegates((Negate) local, (Negate) remote);
    } else if (local instanceof FunctionCall && remote instanceof FunctionCall) {
        return mergeFunctionCall((FunctionCall) local, (FunctionCall) remote);
    } else {
        // If nodes are of the same type but not handled, return local by default
        return local;
    }
}
```

```java
ryOp remoteOp) {
inaryOp remoteOp) {
```

```java
@Test  TeoField-Marsham +1
public void longerBinaryOperations() {
    Formula formula1 = createFormula( expression:
    Formula formula2 = createFormula( expression:
    mergeResult = crdtMerge.merge(formula1, for
    Assertions.assertEquals( expected: "((18*A6)/
}
```

Universität St.Gallen

Bachelor Project

Thank you for your attention!