The Weird and Wonderful World of Excel
A PL Perspective
• 300m active users
• Makes Excel the #1 most used declarative programming language in the world
What?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>=SIN(A2)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>=A3+A4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

A1=SIN(A2)
A2=A3+A4
A3=2
A4=5
Formula Language

Syntax
Recap of Basic Syntax

• Formulas
  • Operations
  • Function Calls
• Ranges
• Arrays
Advanced Syntax

• Cross-sheet references (a.k.a 3d references)
• Array-entered formulae (more on this later…)
• Range operations
• Optional[/missing] arguments
Formula Language

Semantics
Type System!

• 4 Basic Types (that can live in cells)
  • Numbers
  • Strings
  • Booleans
  • Errors

• 2 vector types (that cannot live in cells)
  • Ranges
  • Arrays
Types...

• All operations and functions have type signatures
  • (+) :: N, N -> N

• Not all so simple, functions have:
  • Positional arguments
  • Repeated argument group
  • Return type

• Typical:
  • SUM :: N; N -> N  (note the difference with (+))
Types... (cont)

• "3"+"4"  
  - (+) :: N, N; -> N  
  - "3" :: S, "4" :: S  

• Returns 7  
  - S can be coerced to N
Coercions

• N, S, B can be coerced between each other
• Ranges can be coerced to N, S, B
• Arrays do not coerce, they *lift*
  • \( \text{SIN}\{\{1,2,3\}\} = \{\text{SIN}(1), \text{SIN}(2), \text{SIN}(3)\}\)
  • Multiple unexpected arrays zip:
    • \{1, 2\} + \{3, 4\} = \{1 + 3, 2 + 4\}
• Errors cannot be coerced, they *propagate*...
Errors

• **not** “exceptional” values
• Passing an unexpected error to a function will *not* call the function
  • Instead it will immediately return the error unchanged
Ranges

• Many types:
  • A1 (cell reference)
  • A1:B2 (2d range)
  • A1 B2 (intersection)
  • (A1, A2) (union)
  • Sheet1:Sheet2!A1:B2 (3d)
Ranges... (cont)

• Unexpected ranges:
  • Get dereferenced
    • If range is single row/col then pick intersection with home cell
    • Else-if range is on a different sheet, intersect with home cell across sheets
    • Else pick top-left corner
Excel

Implementation
Calc

• “Push” dependency model
  • Dependency graph is forest of DAGs pointing to dependents

• Any change will propagate forward through the dependencies
• Each DAG can be computed in parallel\(^1\)

\(^1\)Too complex to talk about here
Array-entered Formulae

- `[ctrl]-[shift]-[enter]`
  - Inserts a single formula over many cells

\[
\begin{array}{c|c}
\hline
 & A & B \\
\hline
1 & =\{1;2\} & \\
2 & =\{1;2\} & \{A1:A2=\{1;2\}\} \\
3 & & \\
4 & & \\
\hline
\end{array}
\]
Array-entered... (cont)

- Problem? Overlapping regions
  - \{A1:A2=A2:A3+1\}

- Excel is fine with this!
Disclaimer

- Source from OpenOffice document:
  - The Microsoft Excel File Format
Floats

- Not IEEE-754 compliant
  - No Infinities
  - No NaN
  - No subnormal numbers
- Truncation
Floats… (cont)

• =SUM(0.1, 0.1, 0.1)
  • IEEE 754 representation would make this ~0.3000000000444
  • But =SUM(0.1, 0.1, 0.1)=0.3 is TRUE

• =(SUM(0.1, 0.1, 0.1) – 0.3) = 0
  • This is FALSE
XLS File Format

• Binary file format
  • Multiple *streams* of data arranged hierarchically
    • Workbook stream
      • Globals stream
      • Worksheet stream
        • Cell stream
          • Compiled formula
Formula Compilation

1. Formula GetsParsed (this alone is enough for an hours rant)
2. Formula Gets Compiled to Bytecode (!)
3. Bytecode gets interpreted by Excel
Bytecode

• Way too much to cover here
• A small sample of fun looking instructions:
  • PtgAttrGoto
  • PtgAttrSpace
  • PtgAttrSpaceSemi
  • PtgAttrSpaceType
  • PtgInt
  • PtgNum
  • PtgElfRadicalLel (don’t ask…)
Bytecode… (cont)

- =1+1
  - Compiles to:

<table>
<thead>
<tr>
<th>PtgInt 1</th>
<th>PtgInt 1</th>
<th>PtgAdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x05</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>0x05</td>
<td>0x01</td>
<td>0x03</td>
</tr>
</tbody>
</table>
Bytecode… (cont)

• \(=1.0+2.0\)
  • Compiles to:

<table>
<thead>
<tr>
<th>PtgNum 1.0</th>
<th>PtgNum 1.0</th>
<th>PtgAdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Bytes</td>
<td>9 Bytes</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

• 17 Bytes!
  • An ASCII encoding would only be 8 …
Back to floats

• \(=( \text{SUM}(0.1, 0.1, 0.1) - 0.3)\)

<table>
<thead>
<tr>
<th>PtgNum 0.1</th>
<th>PtgNum 0.1</th>
<th>PtgNum 0.1</th>
<th>PtgFunc SUM</th>
<th>PtgNum 0.3</th>
<th>PtgSub</th>
<th>PtgParen</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 bytes</td>
<td>9 bytes</td>
<td>9...</td>
<td>3...</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

• 41 bytes in total!

• If a formula bytecode ends in PtgAdd or PtgSub it truncates the value written to the cell
Parentheses in bytecode?

=IF(TRUE, 1, 2)

• Compiles to

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>PtgBool TRUE</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>PtgAttrIf 11</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>PtgAttrSpace 1</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>PtgInt 1</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>PtgAttrGoto +18</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>PtgAttrSpace</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>PtgInt 2</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>PtgAttrSpace 1</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>PtgAttrGoto 3</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>PtgFunc IF</td>
</tr>
</tbody>
</table>
End