

Software Interlocking Blocks

SWIBs and Receptors and Black Boxes and the Future of Programming

Paul Tarvydas, May 2024

Goals

- ❖ Simple Expressivity for Programmers, incl. Systems Programmers
- ❖ Breaking Free of the Function-Based Paradigm
- ❖ Abstraction, Namespaces, Concurrency, Black Boxes, etc., etc.
- ❖ Multiple notations, not just a single notation.
- ❖ Hardware focus instead of functional notation focus
- ❖ LEGO®-like software blocks
- ❖ Compiling diagrams to code is easy, why stick to text-only notations?

Textual Notation

- ❖ Excellent for expressing computation, *i.e. calculators*
- ❖ No need to replace text with any other kind of notation
- ❖ Too restrictive for non-computation, *e.g. distributed computing, internet, robotics, blockchain, etc.*
- ❖ FoP instead of FoPoP ... wanted - *Future of Programming vs. Future of Past of Programming ; different problems (biases) in Past than Now and Future (old fashioned ideas about time-sharing, memory sharing vs. today's inexpensive CPUs and inexpensive memory)*

Textual Notation - Restrictions

- ❖ Ad-hoc blocking - *caller must wait for callee*
- ❖ Routing - *callee must send value back to caller, only*
- ❖ Excessive coupling - *functions call other functions (instead of sending messages)*
- ❖ 1:1 - 1 input, 1 output - *destructuring only creates the illusion of multiple inputs & outputs*

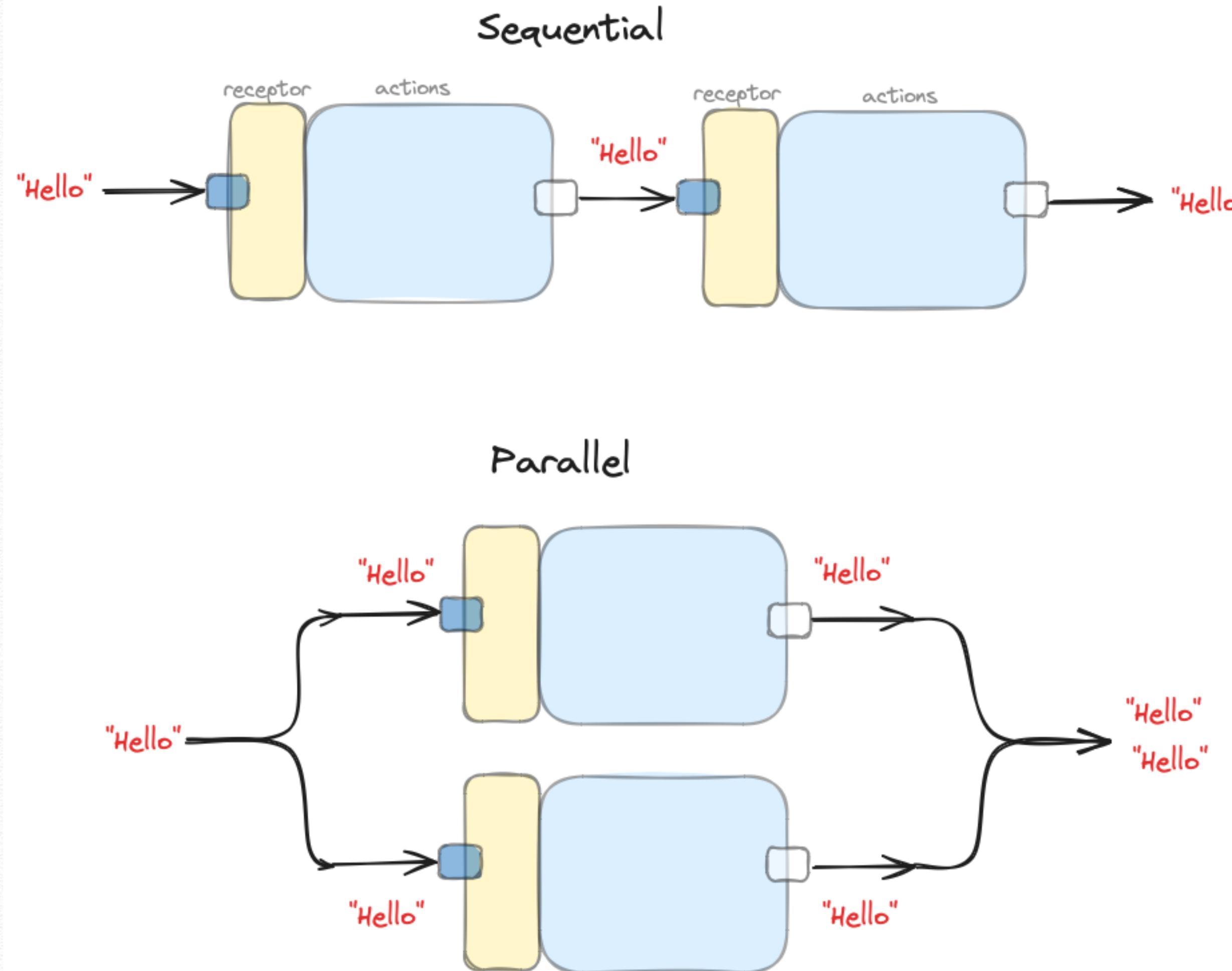
Textual Notation - Restrictions (2)

- ❖ Everything included in single notation - *GPL*
- ❖ Time dimension ignored - *how to express sequencing instead of computing?*
- ❖ Ignores hardware in lieu of notation worship - *hardware can do more than textual notation allows for*
- ❖ Text-based “async” is not true async - *step-wise simultaneity is not asynchronous*
- ❖ Backwards - *notation for analysis drives programming, instead of programming driving analysis*

Textual Notation - Restrictions (3)

- ❖ Single technique for abstraction - *lambdas and parameterization - everything looks the same regardless of semantic importance - prefer to use different notations appropriate to each semantic concern*
- ❖ Inheritance OK for expressing structure of data
- ❖ Inheritance Not OK for expressing control flow - *need parental authority, not inheritance, children must not override behaviour of parents ; original intent of COND was to express conditional values of functions, not as a mixin with variables to create varied control flows*
- ❖ Original intent of CPU hardware was single threading - use multiple CPUs if multiple threads needed

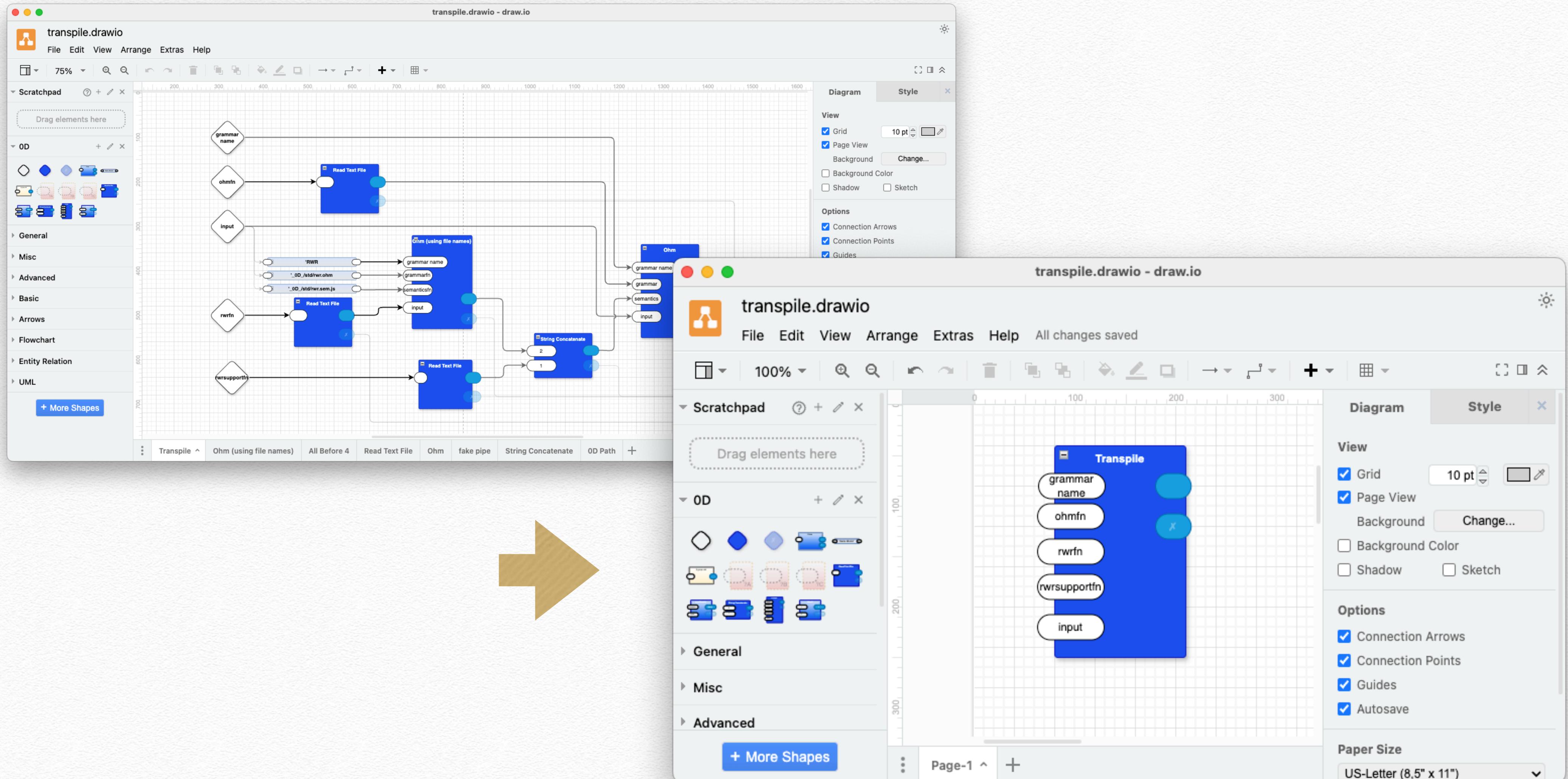
Concurrency is Simple



Concurrency is Simple Continued

- ❖ Don't need thread libraries
- ❖ Anonymous functions, e.g. closures, are enough
- ❖ FIFO queues instead of LIFO callstack for concurrency
- ❖ Loops & recursion don't make sense in async paradigm

Abstraction



Abstraction (2)

- ❖ Abstraction requires fan-out *avoided when analysis drives programming*
- ❖ Enables layered design

Prerequisites for SWIBs

- ❖ Message passing that doesn't use the callstack
- ❖ Hierarchical, recursive message dispatcher (0D)
- ❖ Extreme Isolation - data isolation *and* control-flow isolation

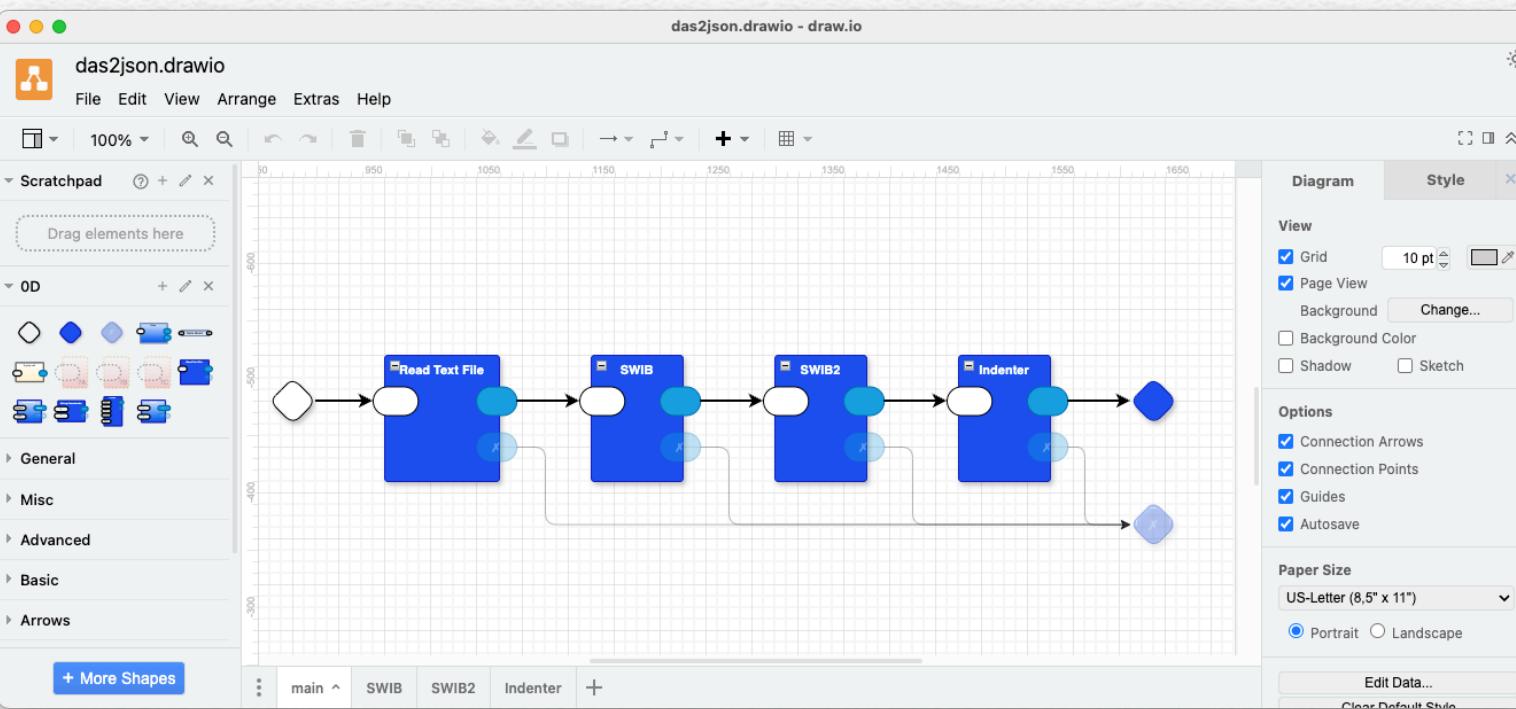
Designing DPLs

- ❖ Choose a small subset of simple graphics, akin to using only a subset of English for textual programming languages
- ❖ Rectangles, ellipses, arrows, text, grouping
- ❖ If editor does not provide convenient semantic info, use simple math and infer the semantic info
 - ❖ e.g. *intersection*, *containment*, *bigger*, *smaller*, *left-of*, etc. for-loops, dictionaries, factbases, Prolog, etc.

Compiling SWIBs to Code

- ❖ Use graphic editors that save diagrams in text format, e.g. XML, graphML
- ❖ Use text parsing tools to parse XML, e.g. OhmJS
- ❖ Feed parse to existing languages, e.g. Python, then let existing language compiler do the rest of the work
- ❖ t2t - text to text translation using PEGs *or* macros
 - ❖ e.g. XML  Python, *or*, XML  .json  diagram interpreter

Example: Diagram Compiler



```

<mxfile host="Electron" modified="2024-05-07T12:09:25.599Z" agent="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.6010.136 Safari/537.36" version="250">
<diagram name="main" id="m0zTKYATKHEmB38VwNw">
<graphModel dx="250" dy="1774" gridSize="10" guides="1" connect="1" arrows="1" fold="1" page="1">
<root>
<mxCell id="g1" parent="0" />
<mxCell id="i1" parent="g1" />
<mxCell id="v0ysikEvkDtxCklj3-15" value="Read Text File" style="rounded=1;whiteSpace=wrap;sketch=0;container=1;" recursiveRe="0">
<xGeometry x="-98" y="-520" width="100" height="110" as="geometry">
<mxRectangle x="-98" y="-520" width="100" height="110" as="alternateBounds" />
</xGeometry>
<mxGeometry x="-18" y="27.5" width="39.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-17" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="-18" y="27.5" width="39.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-17" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="88" y="27.5" width="35.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-18" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="88" y="67.5" width="35.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-19" value="SWIB" style="rounded=1;whiteSpace=wrap;sketch=0;container=1;recursiveRe="0">
<xGeometry x="114" y="520" width="80" height="110" as="geometry" />
<mxRectangle x="114" y="520" width="80" height="110" as="alternateBounds" />
</mxCell>
<mxCell id="v0ysikEvkDtxCklj3-20" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="-10" y="27.5" width="39.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-21" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="60" y="27.5" width="35.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-22" value="" style="rounded=1;whiteSpace=wrap;sketch=0;points=[[0,0.5,0,0,0],[1,0,0,0,0]]" recursiveRe="0">
<xGeometry x="60" y="67.5" width="35.25" height="25" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-35" value="" style="rhombus;whiteSpace=wrap;rounded=1;fontStyle=1;glass=0;sketch=0" recursiveRe="0">
<xGeometry x="1610" y="500" width="40" height="40" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-36" value="" style="rhombus;whiteSpace=wrap;rounded=1;fillColor=#0050ef;fontColor=white" recursiveRe="0">
<xGeometry x="1610" y="-392.5" width="40" height="40" as="geometry" />
<mxCell id="l01_wM0yotGc7zeb" value="edgeStyle=orthogonalEdgeStyle;rounded=0;orthogonalLoop=1;jettySize:auto;" recursiveRe="0">
<xGeometry relative="1" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-42" value="" style="rhombus;whiteSpace=wrap;rounded=1;fontStyle=1;glass=0;sketch=0" recursiveRe="0">
<xGeometry relative="1" as="geometry" />
<mxCell id="v0ysikEvkDtxCklj3-44" value="" style="edgeStyle=orthogonalEdgeStyle;curved=0;rounded=1;orthogonalLoop=1;jettySize:auto;" recursiveRe="0">
<Array>
<xPoint x="1260" y="440" />
<xPoint x="1260" y="-372" />
</Array>
</mxCell>
<mxCell id="v0ysikEvkDtxCklj3-45" value="" style="edgeStyle=orthogonalEdgeStyle;curved=0;rounded=1;orthogonalLoop=1;jettySize:auto;" recursiveRe="0">
<Array>
<xPoint x="1100" y="440" />
<xPoint x="1100" y="-372" />
</Array>
</mxCell>

```

*OhmJS
OhmJS+rwr
etc.*

```

def Das2json (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("Das2json")
    XML (_r)
    _r.append_returned_string ()
    Spaces (_r)
    _r.append_returned_string ()
    _r.eof ()
    _r.end_breadcrumb ("Das2json")
    return _r.return_string_pop ()

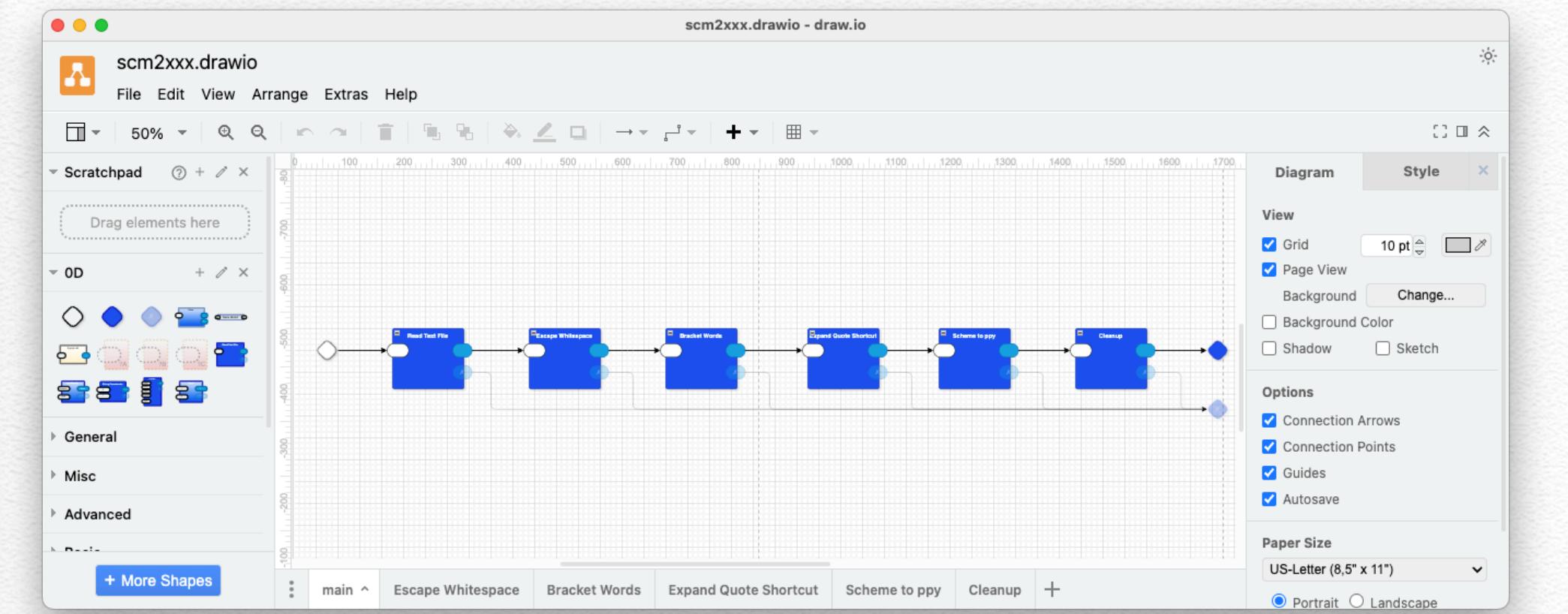
def XML (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("XML")
    Spaces (_r)
    _r.append_returned_string ()
    _r.need_and_append ("<")
    Stuff (_r)
    _r.append_returned_string ()
    if False:
        pass
    elif _r.maybe_append (">"):
        Content (_r)
        _r.append_returned_string ()
        _r.need_and_append ("</")
        Stuff (_r)
        _r.append_returned_string ()
        _r.need_and_append (">")
        pass
    elif _r.maybe_append (">/"):
        pass
    _r.end_breadcrumb ("XML")
    return _r.return_string_pop ()

def Content (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("Content")
    while True:
        Spaces (_r)
        _r.append_returned_string ()
        if False:
            pass
        elif _r.peek ("</"):
            break
            pass
        elif _r.peek ("<"):
            XML (_r)
            _r.append_returned_string ()
            pass
        elif True:
            Stuff (_r)
            _r.append_returned_string ()
            pass
    _r.end_breadcrumb ("Content")
    return _r.return_string_pop ()

def Attributes (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("Attributes")
    while True:
        Spaces (_r)
        _r.append_returned_string ()
        if False:
            pass
        elif _r.peek ("<"):
            XML (_r)
            _r.append_returned_string ()
            pass
        elif True:
            Stuff (_r)
            _r.append_returned_string ()
            pass
    _r.end_breadcrumb ("Attributes")
    return _r.return_string_pop ()

```

Example: Diagram Interpreter



```
scm2xxx.drawio
<mxfile host="Electron" modified="2024-04-28T01:26:58.440Z" agent="Mozilla/5.0 (Macintosh; Intel...>
<diagram name="main" id="m0zTKrYATkNEmBja8Vew">
<mxGraphModel dx="1190" dy="1548" gridSize="10" guides="1" tooltips="1" connect="1">
<root>
<mxCell id="0" />
<mxCell id="1" parent="0" />
<mxCell id="vDygsiKEvxKdtxCckLj3-15" value="Scheme to ppy" style="rounded=1;whiteSpace=w...>
<mxGeometry x="1180" y="-520" width="130" height="110" as="geometry">
<mxRectangle x="-98" y="-1230" width="99" height="26" as="alternateBounds" />
</mxGeometry>
<mxCell id="vDygsiKEvxKdtxCckLj3-16" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="-10" y="27.5" width="39.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-17" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="27.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-18" value="x" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="67.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-19" value="Cleanup" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="1430" y="-520" width="130" height="110" as="geometry">
<mxRectangle x="-98" y="-1230" width="99" height="26" as="alternateBounds" />
</mxGeometry>
<mxCell id="vDygsiKEvxKdtxCckLj3-20" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="-10" y="27.5" width="39.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-21" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="27.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-22" value="x" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="67.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-23" value="Bracket Words" style="rounded=1;whiteSpace=w...>
<mxGeometry x="680" y="-520" width="130" height="110" as="geometry">
<mxRectangle x="-98" y="-1230" width="99" height="26" as="alternateBounds" />
</mxGeometry>
<mxCell id="vDygsiKEvxKdtxCckLj3-24" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="-10" y="27.5" width="39.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-25" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="27.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-26" value="x" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="67.5" width="35.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-27" value="Escape Whitespace" style="rounded=1;whiteSpa...>
<mxGeometry x="430" y="-520" width="130" height="110" as="geometry">
<mxRectangle x="-98" y="-1230" width="99" height="26" as="alternateBounds" />
</mxGeometry>
<mxCell id="vDygsiKEvxKdtxCckLj3-28" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="-10" y="27.5" width="39.25" height="25" as="geometry" />
</mxCell>
<mxCell id="vDygsiKEvxKdtxCckLj3-29" value="" style="rounded=1;whiteSpace=wrap;sketch=0;>
<mxGeometry x="110" y="27.5" width="35.25" height="25" as="geometry" />
</mxCell>
```

WIP <https://github.com/guitarvydas/scm2js0d>

JSON

```
scm2xxx.drawio.json
{
  "file": "scm2xxx.drawio",
  "name": "main",
  "children": [
    {
      "name": "Escape Whitespace",
      "id": 3
    },
    {
      "name": "Cleanup",
      "id": 6
    },
    {
      "name": "Bracket Words",
      "id": 10
    },
    {
      "name": "Read Text File",
      "id": 18
    },
    {
      "name": "Scheme to ppy",
      "id": 22
    },
    {
      "name": "Expand Quote Shortcut",
      "id": 27
    }
  ],
  "connections": [
    {
      "dir": 0,
      "source": {
        "name": "",
        "id": 0
      },
      "source_port": "",
      "target": {
        "name": "Read Text File",
        "id": 18
      },
      "target_port": ""
    },
    {
      "dir": 1,
      "source": {
        "name": "Read Text File",
        "id": 18
      },
      "source_port": "",
      "target": {
        "name": "Escape Whitespace",
        "id": 3
      },
      "target_port": ""
    }
  ]
}
```

odin0D
py0D

Das2json Compiler

das2json.swib

```
swib {
    main = spaces rule rule*
    rule = ":" spaces ruleName "=" spaces pattern+
    pattern = stringMatch | endop | cond | peekcond | cycle | rulecall

    endop = "_end" spaces

    ruleName = name spaces

    rulecall = name spaces

    stringMatch = string spaces

    string = dq notdq* dq
    dq = "\""
    notdq =
        | "\\" any -- escaped
        | ~dq any -- raw

    cond = "[" spaces condClause+ "]"
    condClause = "|" spaces peekCondClause+
    peekCondClause = "[" spaces peekCondMatch ":" spaces action*
    condMatch = "| spaces condMatch ":" spaces action*
    condMatch =
        | string -- string
        | endop -- endop
        | "*" -- else
    peekCondMatch =
        | string -- string
        | endop -- endop
        | "*" -- else
    action = break | acceptAndAppend | pattern
    break = "_break" spaces
    acceptAndAppend = "." spaces

    cycle = "<<<" spaces pattern+ ">>>" spaces

    name = firstLetter moreLetter*
    firstLetter = letter | "_"
    moreLetter = digit | firstLetter
}
```

```
swib {
    main [spaces firstRule moreRule*] = '«firstRule»«moreRule»

import receptor
    receptor.Receptor ()
[firstRule] {_.r}
s = '_r.pop_return_value ()'
print (s)
'
rule [_colon spaces rulename _eq spaces2 pattern+] =
def «rulenam» (_r):
    _r.push_new_line ()
    _r.push_breadcrumb ("«rulenam»")
    _r.push_breadcrumb ("«rulename»")
    pattern=_r.endBreadcrumb ("«rulename»")
    return _r.return_string_pop ()-\n'

pattern [p] = '«p»'
endop [_end spaces] = '_r.eof ()\n'
rulenam [name spaces] = '«name»'
rulecall [name spaces] = «name» (_r)\n_r.append_returned_string ()\n'
stringMatch [s ws] = '_r.need_and_append («s»)\n'

string [dq1 notdq* dq2] = "'«notdq»"
dq [q] = '«q»'
notdq_escaped [_bs c] = '«_bs»«c»'
notdq_raw [c] = '«c»'

cond [lb spaces1 condClause+ rb spaces2] = 'if False:(_\nnpass_)«condClause»'
condClause [bar spaces1 condMatch _colon spaces2 action*] = 'elif «condMatch»:(_\naction»
condMatch_string [x] = '_r.maybe_append («x»)'
condMatch_endop [x] = '_r.eof ()'
condMatch_else [x] = 'True'

peakCond [lb spaces1 condClause+ rb spaces2] = 'if False:(_\nnpass_)«condClause»'
peakCondClause [bar spaces1 condMatch _colon spaces2 action*] = 'elif «condMatch»:(_\nac»
peakCondMatch_string [x] = '_r.peek («x»)'
peakCondMatch_endop [x] = '_r.eof ()'
peakCondMatch_else [x] = 'True'

action [a] = '«a»'
break [_break spaces] = 'break\n'
acceptAndAppend [_dot spaces] = '_r.accept_and_append ()\n'

cycle [lb spaces1 pattern+ rb spaces2] = 'while True:(_\npattern_)«`n»

name [firstLetter moreLetter*] = '«firstLetter»«moreLetter»'
firstLetter [c] = '«c»'
moreLetter [c] = '«c»

}

U:--- swib.rwr      All L37  (Fundamental)
Beginning of buffer
```

The screenshot shows a text editor interface with two tabs open:

- defname.ohm**: The top tab contains the following text:

```
defname {
    main = text+
    text =
        | defName -- match
        | any -- other
    defName = "/" spaces "def" spaces name spaces through<closeindent> spaces ")"
    through<s> = (~s any)+ s
    name = letter alnum* ~alnum
    closeindent = "-")"
```
- defname.rwr**: The bottom tab contains the following text:

```
defname {
    main [x+] = '«x»'
    text_match [x] = '«x»'
    text_other [x] = '«x»'
    defName [lb spaces1 _def spaces2 name spaces3 misc spaces4 rb] = '«name»'
    through [misc s] = ''
    name [letter alnum*] = '«letter»«alnum»'
    closeindent [rb] = ''
}
```

The screenshot shows a BPMN-like process diagram in the draw.io application. The diagram consists of five main nodes connected by arrows:

- A start node (diamond) connected to a "Read Text File" activity.
- "Read Text File" activity connected to a "SWIB" activity.
- "SWIB" activity connected to a "SWIB2" activity.
- "SWIB2" activity connected to an "Indenter" activity.
- "Indenter" activity connected to an end node (diamond).

There are also two parallel gateway nodes (hexagons) in the diagram. One gateway is positioned between the "Read Text File" and "SWIB" activities, with a connector line branching off to the right. Another gateway is positioned between the "Indenter" and end node, with a connector line branching off to the left. The entire diagram is set against a grid background with numerical coordinates ranging from 850 to 1650 on both the x and y axes.

OD

The image features three dark green arrows on a light beige background. A large, solid dark green downward-pointing arrow is centered at the top. Below it, on the left and right sides, are two smaller dark green right-pointing arrows. The arrows have a subtle diagonal hatching pattern.

```
: Das2json =
XML Spaces _end

: XML =
Spaces "<" Stuff
[
| ">": Content "</>" Stuff ">"
| "/>":
]

: Content =
<<<
Spaces
[*|
| "</>": _break
| "<": XML
| *: Stuff
]
>>>

: Attributes =
<<<
[*|
| ">": _break
| "/>": _break
| _end: _break
| *: Stuff
]
>>>

: Stuff =
<<<
[*|
| ">": _break
| "<": _break
| "/>": _break
| _end: _break
| *: .
]
>>>

: Spaces =
<<<
[*|
| " ": .
| "\t": .
| "\n": .
| *: _break
]
>>>

: String =
"\\" NotDquotes \""
: NotDquotes =
:---- das2json.swib Top L49 Git:main (Fundamental)
```

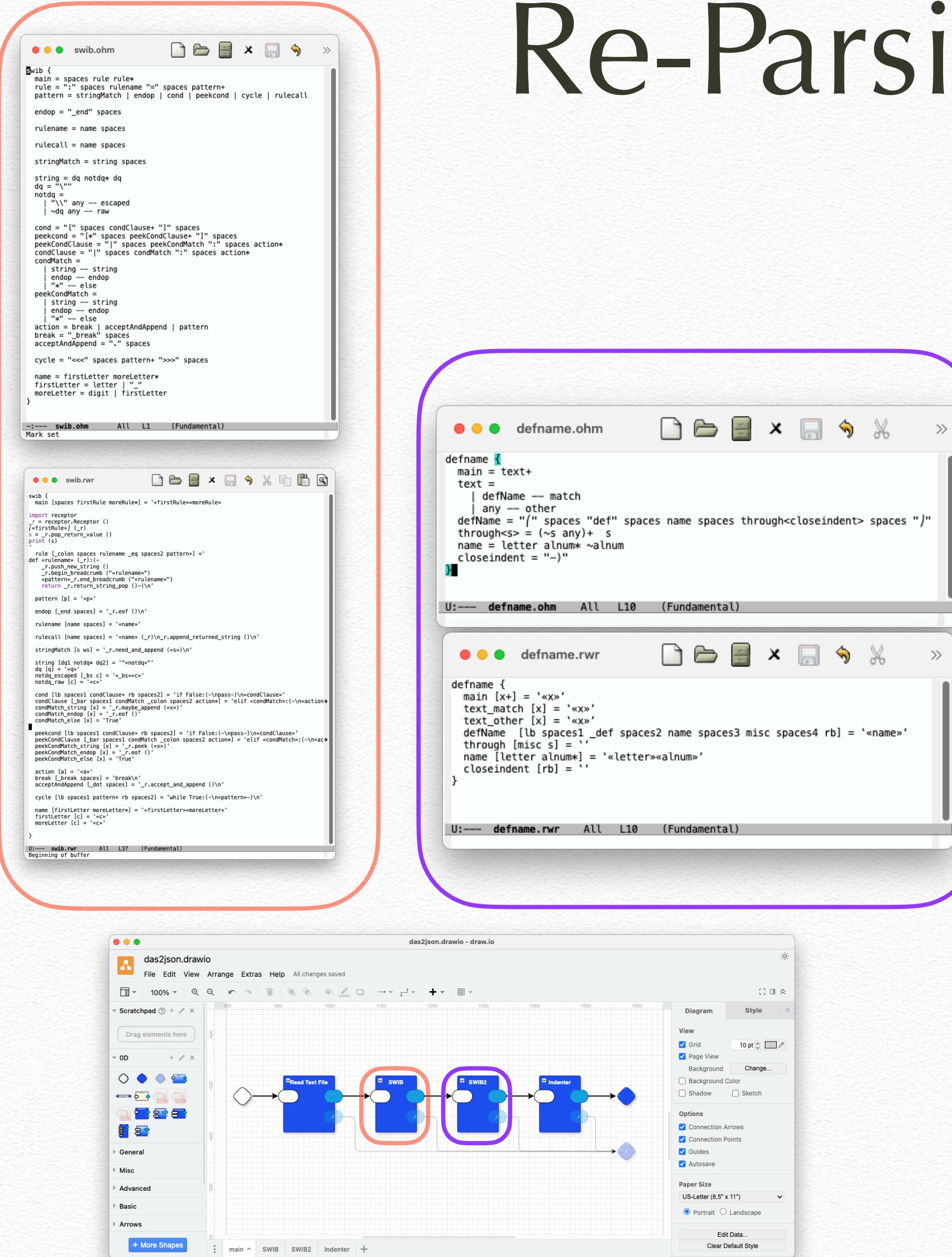
das2json.py

```
Das2json (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("Das2json")
XML (_r)
    _r.append_returned_string ()
Spaces (_r)
    _r.append_returned_string ()
    _r.eof ()
    _r.end_breadcrumb ("Das2json")
    return _r.return_string_pop ()

XML (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("XML")
Spaces (_r)
    _r.append_returned_string ()
    _r.need_and_append ("<")
Stuff (_r)
    _r.append_returned_string ()
if False:
    pass
elif _r.maybe_append (>):
    Content (_r)
    _r.append_returned_string ()
    _r.need_and_append ("</")
    Stuff (_r)
    _r.append_returned_string ()
    _r.need_and_append (>)
    pass
elif _r.maybe_append (/>):
    pass
    _r.end_breadcrumb ("XML")
    return _r.return_string_pop ()

Content (_r):
    _r.push_new_string ()
    _r.begin_breadcrumb ("Content")
while True:
    Spaces (_r)
    _r.append_returned_string ()
    if False:
        pass
    elif _r.peek (</):
        break
        pass
    elif _r.peek (<):
        XML (_r)
        _r.append_returned_string ()
        pass
    elif True:
        pass
```

Re-Parsing Generated Code



Need to extract info (*name* in this example) from generated code - use 2nd parser instead of special-case data structures and return values.

WIP Projects

- ❖ Kinopio to markdown (“mind-map” automatically reorganized as markdown (for ChatGPT?))
- ❖ Dungeon Crawler game prototype (based on example game in Ceptre paper)
- ❖ Arith0D - grammar snippet compiled simultaneously to Python/Javascript/Common Lisp/WASM
- ❖ Visual SHeLL - VSH
- ❖ Scheme to Javascript transpiler, Scheme to Python transpiler
- ❖ LLM wrapper
- ❖ SWIB generator from diagrams
- ❖ new syntax for find-and-replace using HTML markup for colorization and fontification
- ❖ Invention of new streaming parser language Receptor for SWIBs (das2json compiler)