The Co-dfns Compiler

Aaron W. Hsu

June 8, 2022

Contents

1 Introduction

2 User's Guide

3 Co-dfns Architecture

This section describes the "big picture" parts of the Co-dfns compiler. The intent here is to try to show how all of the various moving parts of the compiler fit together, to provide a sort of road map that will give you a precise plan for understanding how the various components affect one another. One of the most important things to understand in any compiler is the net effect a local change in the code can have on the rest of the system, so I hope that this section will help to clarify this.

The design of the Co-dfns compiler is one of austerity and minimalism. My intent is, was, and hopefully shall remain that of producing an exceptionally clear design that avoids or eliminates unnecessary code and complexity within the design. I attack this problem in many ways, but I primarily attempt to do this by both reducing the size of the code surface in total, that is, write less code, as well as reducing the number of entry points and paths through that code. In other words, my ideal design is one in which you enter the compiler in some limited, but well defined and useful set of entry points, and then proceed in a linear fashion through the code as the execution path, resulting finally in your result. This is the "ultimate" in data flow, functionally oriented programming.

The ramifications of this design choice implies a few important things. Firstly, it implies that I reduce and eliminate any code that represents boilerplate or that does not actively contribute to the "big picture" of the code. This is required in an extreme degree if I am to reduce the overall complexity of the design. This also implies that there is very little intentional redundancy in the shape and style of the source, making it very terse and compact. Since there are intentionally very few entry and exit points through the control flow of the code, this reduces the number of dependencies for me to be aware of when dealing with a single piece of code, but this also comes at the cost of not being able to see many examples of the interfaces with that code. Often, there will be one, and only one place, in which a given piece of code is used, and I do not want the code to needlessly store excess information in its source that doesn't need to be there.

This all culminates in something that can be quite shocking at first: making a change to the source is almost always a big deal. If all the source code is meaningful and carefully constructed, this also means that making changes to this code are almost always non-trivial,

because if the code represented something trivial, I would have tried to remove it from the code so that only the "big things" were in the code itself. Thus, anyone who wishes to view and read the compiler code should take it upon themselves to appreciate the way in which the code flows together, and how the flow of the program runs, as doing so will be essential to understanding how to make changes to the source without breaking something. Fortunately, this does come with the intended benefits of a very short and simple codebase that has clear flow through the system, it just means that if you want to change something, make sure you realize that you are almost always likely to be working at the "architectural" level, rather than at the small and trivial level of details.

The compiler is designed to fit into a single Dyalog APL namespace, and importantly, we do not define additional nested namespaces or other forms of name hiding. I intentionally want to restrict the namespace to a single global one. This single global namespace should therefore contain the carefully curated names that matter, and any that do not matter should, ideally, not be defined or used. The namespace itself can be divided into three main groupings: the public facing entry-points into the system, the compiler logic itself, and the utilities or other elements that serve to support the others. This gives use the following code outline.

4 ⟨* 4⟩≡

```
:Namespace codfns

      ⟨Global Settings 6a⟩
      ⟨The Fix API 9⟩
      ⟨User-command API 11a⟩

      ⟨Parser 13⟩
      ⟨Compiler 23⟩
      ⟨Code Generator 24⟩
      ⟨Interface to the backend C compiler 29⟩
      ⟨Linking with Dyalog 30a⟩

      ⟨AST Record Structure 11b⟩
      ⟨Converters between parent and depth vectors 11c⟩
      ⟨XML Rendering 34⟩
      ⟨Pretty-printing AST trees 31c⟩

:EndNamespace
```

Root chunk (not used in this document).
Defines:
codfns, used in chunks 24, 29, 35, 38, 40, and 41.

The primary user-facing interfaces into the compiler are ⟨*The Fix API* 9⟩ and the ⟨*User-command API* 11a⟩. These are the ways that you primarily drive the entire compiler. I intentionally expose the rest of the compiler interfaces without hiding them so that people who wish to leverage these other parts of the system without using the "entire" compiler pipeline are able to do so, but I do not consider this a public interface.

This distinction matters because of our testing philosophy and our version numbering. Generally speaking, our version numbering scheme only tracks a major or minor change in the compiler when the externally facing interfaces receive some fundamental changes. Changes to the internal changes are *not* considered for this versioning scheme. Moreover, since I intend for there to be great freedom in changing and altering the behavior of these internal pipeline interfaces, these interfaces are not directly tested, and the test suite should *not* include testing against these internal interfaces. We philosophically only test against the external interfaces, and eschew internal unit tests.[1]

The utility functions defined below the core compiler pipeline represent functionality that is tangential to the main compiler operation. However, these utilities also tend to represent some specific insight into the design of the compiler. Understanding the core AST structure and design as well as getting a grip on how to manipulate the core tree manipulation structures are vital to understanding the rest of the code. Therefore, this section spends more time on discussing these topics before the upcoming sections dealing with a more detailed exposition of the compiler itself. However, there are utilities that we consider more advanced, such as the pretty-printing functions and XML rendering that are topics of interest to advanced users of the compiler, but which are not part of the main compiler pipeline. Even though these functions have intentionally general application and are likely to be useful not only to those working on the compiler itself but also to those who are using more advanced compiler features, these utilities are not critical to a deep understanding of the compiler, so these are not discussed in this section. Instead, we discuss those topics in the section on developer tooling and infrastructure concerns.

The remaining parts of this section will describe the external facing interfaces to the compiler as well as the core underlying data structures and idioms that form the underlying skeleton and foundation for writing and working with any aspect of the compiler. These are all feature and component agnostic elements of the system that do not belong solely to only a single part, but that impact all other

[1] You can read more of my opinions on this matter in my article, "The Fallacy of Unit Testing".

elements of the compiler source code, and so it pays especially well to pay attention and understand this code to a high degree.

3.1 Global Settings

There are some global options that we assume to exist throughout the compiler. These set the standard behaviors as well as serve as knobs that can be tweaked in some cases to identify what behaviors we want from the rest of the compiler.

First, we have a set of read-only global constants that are defined to configure our APL environment. These are the typical ones, and we try to stick to the defaults, except that we are sane, and thus we use ⎕IO set to 0.

6a ⟨*Global Settings* 6a⟩≡

```
⎕IO ⎕ML ⎕WX←0 1 3
```

This definition is continued in chunks 6–8.
This code is used in chunk 4.
Defines:
⎕IO, used in chunk 32.
⎕ML, used in chunk 32.
⎕WX, never used.

Additionally, we set a VERSION constant to track changes to the system through the distributions. We use semantic versioning[2] as our versioning scheme. That being said, we also do not have particular qualms about changing the public API at a rapid pace, provided that we document this.

6b ⟨*Global Settings* 6a⟩+≡

```
VERSION←4 1 0
```

This code is used in chunk 4.
Defines:
VERSION, never used.

[2] https://semver.org/

We depend on ArrayFire[3] for much of our GPU backend functionality. This means we need to know two things, where ArrayFire is installed and which ArrayFire backend we should use when compiling. We only really need to know where ArrayFire is installed on UNIX style systems, as these systems seem to be much more variable in this regard, and there is an environment variable that we can use in Windows to find out where ArrayFire is installed more conveniently on that platform. We default to using `'cuda'` as our main option, but we also support the following options for `AF∆LIB`:

```
cuda opencl cpu
```

Using `''` for `AF∆LIB` will use ArrayFire's unified backend, but we don't default to this because we have seen some issues on some platforms with reliability problems. To avoid this, we choose to use `cuda` as the default, which tends to either work or fail explicitly, which allows the user to respond rather than crashing ungracefully in the case of the unified backend.

The least reliable backend we have seen is the `opencl` one, which seems to be more hit or miss depending on the underlying stability of the OpenCL drivers that are installed on the user's system. In particular, some Linux OpenCL installations seem to be particularly fragile. In such cases, always make sure that a good, solid OpenCL library is being used.

7 ⟨*Global Settings* 6a⟩+≡

```
AF∆PREFIX←'/opt/arrayfire'
AF∆LIB←'cuda'
```

This code is used in chunk 4.
Defines:
`AF∆LIB`, used in chunks 11a, 29, and 40b.
`AF∆PREFIX`, used in chunk 29.

[3] https://arrayfire.com/

On Windows, we rely on the Visual Studio C/C++ compiler to build our runtime and user code. We have settled on trying to stay as up to date with this as possible. However, there are many different installation paths used by Visual Studio, which can make it difficult to know where to look unless we hardcode each location. Instead, we assume that Visual Studio will not be a primary interest to our users, making it likely that they will be installing Visual Studio only as a dependency for using Co-dfns. In this case, it is likely that they will be using the Community version. Thus, we default to using the latest version of Visual Studio of which we are aware and using the Community version of this, which Microsoft does not charge for.

If a different version of Visual Studio is installed, then it is important to figure out what the right path should be to locate the Visual Studio installation. The main thing we need to get from this path is access to the `vcvarsall.bat` batch file. This file configures the `cmd.exe` environment to be able to find the Visual Studio compiler and work in the right way. In the 2002 Community addition, and apparently most new versions of Visual Studio, this is located in the `VC\Auxiliary\Build\` subdirectory of the main installation folder. When changing this path, we want to make sure that the following path points to the correct `vcvarsall.bat` file:

```
VS∆PATH,'\VC\Auxiliary\Build\vcvarsall.bat'
```

Most users will simply need to alter `Community` to match the edition of Visual Studio 2022 that they have installed on their system.

8 ⟨*Global Settings* 6a⟩+≡

```
VS∆PATH←'\Program Files\Microsoft Visual Studio'
VS∆PATH,←'\2022\Community'
```

This code is used in chunk 4.
Defines:
`VS∆PATH`, used in chunks 29 and 40b.

3.2 The Fix API

One of the core entry points into the compiler is through the `Fix` function. This function is designed to mimic and more or less replace the use of the `⎕FIX` function found in Dyalog APL. Its design models that behavior, and it is important as an entry-point because it exercises most of the core elements of the compiler. In particular, the design of the compiler's pipeline is demonstrated most fully in this function.

$$Parse \rightarrow Compile \rightarrow Generate \rightarrow Backend \rightarrow Link$$

The interfaces to the `⎕FIX` function and the Co-dfns `Fix` function differ in a few key ways. The left argument to `Fix` is a character vector giving the name to use when generating files and other artifacts. This does *not* affect the name of the resulting namespace, since that is defined, if at all, in the file source itself. The `⍺` argument only affects the name of the files and other outputs that `Fix` generates.

We also print out which part of the compiler we are in when we enter that "phase". Doing this helps to give us an intuitive sense of how fast each phase is and whether one phase is taking an abnormally long time or not. It also helps in debugging.

9 ⟨*The Fix API* 9⟩≡

```
Fix←{
          _←a n s src←PS ⍵⊣⎕←'P'
          _←           TT _⊣⎕←'C'
          _←           GC _⊣⎕←'G'
          _←         ⍺ CC _⊣⎕←'B'
                     n NS _⊣⎕←'L'
}
```

This code is used in chunk 4.
Defines:
`Fix`, used in chunk 11a.
Uses `src` 40a.

The input requirements for `Fix` are not listed in the definition itself, because both the parser `PS` and the `Fix` function need to use the same basic checks, and since the `Fix` function calls the parser as its first entry point, it doesn't make much sense to duplicate that work in both places. The requirements are as follows:

- Scalar/Vector
- Character type
- Simple or Vector of Vectors

We generate a `DOMAIN ERROR` if the inputs are not well-formed.

10a ⟨*Verify input from* `IN` 10a⟩≡

```
err←'PARSER EXPECTS SCALAR OR VECTOR INPUT'
1<≢⍴IN:err ⎕SIGNAL 11

err←'PARSER EXPECTS SIMPLE OR VECTOR OF VECTOR INPUT'
2<|≡IN:err ⎕SIGNAL 11

⟨Normalize the input formatting 10b⟩

err←'PARSER EXPECTS CHARACTER ARRAY'
0≠10|⎕DR IN:err ⎕SIGNAL 11
```

This code is used in chunk 13.

The input formatting that is accepted means that newlines could be denoted either with `LF`, `CR`, or `CRLF` sequences inside of the vectors themselves or they could be denoted by having separate vectors for the various lines, or even a mixture of both. To simplify this situation we want to normalize them so that we are always dealing with some combination of `LF`, `CR`, and `CRLF` sequences within the file itself, rather than dealing with the nested situation. This ensures that after verification of the input, everything will work off of the same format. We intentionally put a newline at the end of the file even if we may not require one because it is possible that we are dealing with a file that is missing its final newline. By always adding one, we ensure that every line in the input is always terminated by a line ending. Life is also simpler if we just use LF as our line ending instead of something else, this means that future code must be aware that there could be mixed line endings in the file.

10b ⟨*Normalize the input formatting* 10b⟩≡

```
IN←∊(⊆IN),¨⎕UCS 10
```

This code is used in chunk 10a.

3.3 The User Command API

11a ⟨*User-command API* 11a⟩≡

```
∇Z←Help _
 Z←'Usage: <object> <target> [-af={cpu,opencl,cuda}]'
∇

∇r←List
 r←⎕NS¨1⍴⊂⍬ ⋄ r.Name←,¨⊂'Compile' ⋄ r.Group←⊂'CODFNS'
 r[0].Desc←'Compile an object using Co-dfns'
 r.Parse←⊂'2S -af=cpu opencl cuda '
∇

∇ Run(C I);Convert;in;out
⍝ Parameters
⍝      AF⍙LIB         ArrayFire backend to use
 Convert←{⍺(⎕SE.SALT.Load'[SALT]/lib/NStoScript -noname').ntgennscode ⍵}
 in out←I.Arguments ⋄ AF⍙LIB←I.af''⊃¨⍨I.af≡0
 S←(⊂':Namespace ',out),2↓0 0 0 out Convert ##.THIS.⍠in
 →0⍲¨⍨'Compile'≢C
 {##.THIS.⍠out,'←⍵'}out Fix S⍪⎕EX'##.THIS.',out
∇
```

This code is used in chunk 4.
Uses `AF⍙LIB` 7 and `Fix` 9.

3.4 AST Record Structure

11b ⟨*AST Record Structure* 11b⟩≡

```
f⍙←'ptknfsrdx'
N⍙←'ABCEFGKLMNOPSVZ'
```

This code is used in chunk 4.

3.5 Converters between parent and depth vectors

11c ⟨*Converters between parent and depth vectors* 11c⟩≡

```
P2D←{z←⍬⍪⍳≢⍵ ⋄ d←⍵≠,z ⋄ _←{p⍪d+←⍵≠p←⍺[z,←⍵]}⍣≡⍨⍵ ⋄ d(⍋⍨(-1+d)↑⍤0 1⊢⍒z)}
D2P←{0=≢⍵:⍬ ⋄ p⍪2{p[⍵]←⍺[⍺⍳⍵]}⌿⊢∘⊂⍨⍵⍪p←⍳≢⍵}
```

This code is used in chunk 4.

4 Testing

We use the APLUnit testing framework to facilitate our testing of the Co-dfns compiler. The test harness is designed around a testing philosophy in which we ever only write black-box tests that work on the whole compiler using inputs that could be created or are expected to be creatable by end-users. That is, we do no "unit testing" of our source code, but only whole program testing.

The testing framework is provided by the `ut.apln` file, which is not part of this literate program and so is not included in this document. In order to make some of the testing more convenient, we define the function `TEST` to run the tests that exist in the `tests\` subdirectory. Each of these tests has a specific number which defines the test, and we refer to the tests by number when running them. Both of these testing functions assume that we are running inside of the `tests\` directory or one configured identically to it.

The `TEST` function takes either `'ALL'` as its input or a test number in the form of an integer. Given an integer, we call the test matching that number in the current working directory.

The `'ALL'` option causes `TEST` to run all of the tests that are defined in the current working directory. This command is a nicety, since we can technically do all of this by iterating the `TEST` function over the range of test numbers, but this would not create the aggregate statistics that we would like to see at the end of the testing report. By using `'ALL'` we get to see a complete summary of the results of testing all the code, rather than just the individual testing results on a per testing group/number basis.

12 ⟨TEST *Function* 12⟩≡

```
TEST←{
        #.UT.(print_passed print_summary)←1
        'ALL'≡⍵:#.UT.run './'
        path←'./t',(1 0⍕(4⍴10)⊤⍵),'_*_tests.dyalog'
        #.UT.run ⊃⊃0⎕NINFO⍠1⊢path
}
```

Root chunk (not used in this document).
Defines:
`TEST`, used in chunks 13 and 35.

5 Co-dfns Compiler

5.1 Parser

13 ⟨*Parser* 13⟩≡

```
  PS←{IN←ω ⋄ A B C E F G K L M N O P S V Z←1+⍳15
          I←{(⊂ω)⌷α} ⋄ U←{0=⎕NC'α':ωω⍣¯1 αα ωω ω ⋄ ωω⍣¯1⊢(ωω α) αα ωω ω}
          assert←{α←'assertion failure' ⋄ 0∊ω:⍎'α ⎕SIGNAL 8' ⋄ shy←0}

          ⟨Verify input from IN 10a⟩

 ⍝ Line and error reporting utilities
          CR LF←⎕UCS 13 10
          linestarts←(⍸1⍪2>⌿IN∊CR LF)⍪≢IN
          mkdm←{α←2 ⋄ line←linestarts⍸ω ⋄ no←'[',(⍕1+line),'] '
                  i←(~IN[i]∊CR LF)⌿i←beg+⍳linestarts[line+1]-beg←linestarts[line
                  (⎕EM α)(no,IN[i])(' ^'[i∊ω],⍨' 'ρ⍨≢no)}
          quotelines←{
                  lines←∪linestarts⍸ω
                  nos←(1 0ρ⍨2×≢lines)⍀'[',(⍕⍪1+lines),⍤1⊢'] '
                  beg←linestarts[lines] ⋄ end←linestarts[lines+1]
                  m←∊∘ω¨i←beg+⍳¨end-beg
                  ¯1↓∊nos,(~∘CR LF¨⍪,(IN∘I¨i),⍪' ‾'∘I¨m),CR}
          SIGNAL←{α←2 '' ⋄ en msg←α ⋄ EN∘←en ⋄ DM∘←en mkdm ⊃ω
                  dmx←('EN' en)('Category' 'Compiler')('Vendor' 'Co-dfns')
                  dmx,←⊂'Message'(msg,CR,quotelines ω)
                  ⎕SIGNAL⊂dmx}

 ⍝ Group input into lines as a nested vector
          pos←(⍳≢IN)⊆⍨~IN∊CR LF

 ⍝ Mask strings
          0≠≢lin←⍸⊃∘⌽¨msk←≠⍀¨''''=IN∘I¨pos:{
                  EM←'SYNTAX ERROR: UNBALANCED STRING',('S'⌿⍨2≤≢lin),CR
                  EM,←quotelines ∊(msk⌿¨pos)[lin]
                  EM ⎕SIGNAL 2}⍬

 ⍝ Remove comments
          pos msk⌿¨⍨←⊂∧⍀¨(~msk←msk∨¯1⌽¨msk)⍲'⍝'=IN∘I¨pos

 ⍝ Remove leading and trailing whitespace
          WS←⎕UCS 9 32 ⋄ pos msk⌿¨⍨←⊂~(∧⍀∨∧⍀U⌽)∘(WS∊⍨IN∘I)¨pos

 ⍝ Flatten and separate lines and ⋄ with Z type
          t←⊃0ρ⊂pos ⋄ t pos msk(∊,∘⍪⍨)←Z(⊃¨pos)0 ⋄ t[⍸'⋄'=IN[pos]]←Z
```

```
⍝ Tokenize Strings
        end←1+pos ⋄ t[i←⍸2<≠0,msk]←C ⋄ end[i]←end[⍸2>≠msk,0]
        t pos end≠⍨←⊂(t≠0)∨~msk

⍝ Verify that all open characters are part of the valid character set
        alp←'ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz'
        alp,←'ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝßàáâãäåæçèéêëìíîïðñòóôõöøùúûüþ'
        alp,←'ΔΔABCDEFGHIJKLMNOPQRSTUVWXYZ'
        num←⎕D
        synb←'¯[]{}()'':αω⋄;'
        syna←'θ⎕⍞#'
        prmfs←'+-×÷|⌈⌊*⍟○!?~∧∨⍲⍱<≤=>≥≠≡≢⍴,⍪⌽⍉⍋⍒⊂⊆⊃∈⍷∩∪⍳⍸⍷⍋⍒⍪⍉⊥⊤⍲⍱⍞∇←→'
        prmdo←'∘.⌸⌺⍤⍥⍣@' ⋄ prmmo←'¨⍨&⌶⌸' ⋄ prmfo←'/≠\⍀'
        prms←prmfs,prmdo,prmmo,prmfo
        x←' '@{t≠0}IN[pos] ⍝ The spaces produce nice invariants
        v≠msk←~x∈alp,num,syna,synb,prms,WS:{
                EM←'SYNTAX ERROR: INVALID CHARACTER(S) IN SOURCE',CR
                EM,←quotelines ⍳msk
                EM ⎕SIGNAL 2}⍬

⍝ Tokenize numbers
        _←{dm[ω]←∧⍀dm[ω]}¨(dm∨x∈alp)⊆⍳≠dm←x∈num
        dm∨←('.'=x)∧(¯1⌽dm)∨1⌽dm
        dm∨←('¯'=x)∧1⌽dm
        dm∨←(x∈'EeJj')∧(¯1⌽dm)∧1⌽dm
        v≠msk←(dm=0)∧x='¯':2'ORPHANED ¯'SIGNAL pos≠⍨msk
        v≠{1<+≠ω='j'}¨dp←⎕C¨dm⊆x:'MULTIPLE J IN NUMBER'⎕SIGNAL 2
        v≠{1<+≠ω='e'}¨dp←⊃⍪/{ω⊆⍨ω≠'j'}¨dp:'MULTIPLE E IN NUMBER'⎕SIGNAL 2
        v≠'e'=⊃¨dp:'MISSING MANTISSA'⎕SIGNAL 2
        v≠'e'=⊃∘⌽¨dp:'MISSING EXPONENT'⎕SIGNAL 2
        mn ex←⍉⍒⍉{2↑(ω⊆⍨ω≠'e'),⊂''}¨dp
        v≠{1<+≠'.'=ω}¨mn,ex:'MULTIPLE . IN NUMBER'⎕SIGNAL 2
        v≠'.'∊¨ex:'REAL NUMBER IN EXPONENT'⎕SIGNAL 2
        v≠{v≠1↓ω∈'¯'}¨mn,ex:'MISPLACED ¯'⎕SIGNAL 2
        t[i←⍸2<≠0,dm]←N ⋄ end[i]←end≠⍨2>≠dm,0

⍝ Tokenize Variables
        t[i←⍸2<≠0,vm←(~dm)∧x∈alp,num]←V ⋄ end[i]←end≠⍨2>≠vm,0

⍝ Tokenize α, ω formals
        fm←{mm←⌽⊃(>∘⊃,⊢)≠⌽m←α=' ',ω ⋄ 1↓¨(mm∧~m1)(mm∧m1←1⌽m)}
        am aam←'α'fm x ⋄ wm wwm←'ω'fm x
        ((am∨wm)≠t)←A ⋄ ((aam∨wwm)≠t)←P ⋄ ((aam∨wwm)≠end)←end≠⍨¯1⌽aam∨wwm

⍝ Tokenize Primitives, Atoms
```

```
          t[⍸(~dm)∧x∊prms]←P ⋄ t[⍸x∊syna]←A

⍝ Compute dfns regions and type, include } as a child
          t[⍸'{'=x]←F ⋄ 0≠⊃d←¯1⍱+⍀1 ¯1 0['{}'⍳x]:'UNBALANCED DFNS'⎕SIGNAL 2

⍝ Check for out of context dfns formals
         ∨⌿(d=0)∧(t=P)∧IN[pos]∊'⍺⍵':'DFN FORMAL REFERENCED OUTSIDE DFNS'⎕SIGNAL

⍝ Compute trad-fns regions
        ∨⌿Z≠t⌿⍨1⍱msk←(d=0)∧'∇'=x:'TRAD-FNS START/END LINES MUST BEGIN WITH ∇'⎕SIG
NAL 2
          0≠⊃tm←¯1⍱≠⍀(d=0)∧'∇'=x:'UNBALANCED TRAD-FNS'⎕SIGNAL 2
        ∨⌿Z≠t⌿⍨⊃1 ¯1∨.⍱⊂(2>⌿tm)⌿0:'TRAD-FNS END LINE MUST CONTAIN ∇ ALONE'⎕SIGNAL

⍝ Identify Label colons versus others
          t[⍸tm∧(d=0)∧∊((~⊃)∧(<⍀∨⍀))⍤':'=(t=Z)⊂IN[pos]]←L

⍝ Tokenize Keywords
          ki←⍸(t=0)∧(d=0)∧(':'=IN[pos])∧1⍱t=V
          t[ki]←K ⋄ end[ki]←end[ki+1] ⋄ t[ki+1]←0
          ERR←'EMPTY COLON IN NON-DFNS CONTEXT, EXPECTED LABEL OR KEYWORD'
          ∨⌿(t=0)∧(d=0)∧':'=IN[pos]:ERR ⎕SIGNAL 2

⍝ Tokenize System Variables
          si←⍸('⎕'=IN[pos])∧1⍱t=V
          t[si]←S ⋄ end[si]←end[si+1] ⋄ t[si+1]←0

⍝ Delete all characters we no longer need from the tree
          d tm t pos end(≠⌿⍨)←⊂(t≠0)∨x∊'()[]{}:;'

⍝ Tokenize Labels
          ERR←'LABEL MUST CONSIST OF A SINGLE NAME'
          ∨⌿(Z≠t[li-1])∨(V≠t[li←⍸1⍱msk←t=L]):ERR ⎕SIGNAL 2
          t[li]←L ⋄ end[li]←end[li+1]
          d tm t pos end(≠⌿⍨)←⊂~msk

⍝ Now that all compound data is tokenized, reify n field before tree-building
          n←{1↓⍵⍤¨'0',⍵}@{t=N}(⊂'')@{t∊Z F}1 ⎕C@{t∊K S}IN∘I⍤¨pos+⍳⍤¨end-pos

⍝ Verify that keywords are defined and scoped correctly
         KW←'NAMESPACE' 'ENDNAMESPACE' 'END' 'IF' 'ELSEIF' 'ANDIF' 'ORIF' 'ENDIF
        KW,←'WHILE' 'ENDWHILE' 'UNTIL' 'REPEAT' 'ENDREPEAT' 'LEAVE' 'FOR' 'ENDFOR
          KW,←'IN' 'INEACH' 'SELECT' 'ENDSELECT' 'CASE' 'CASELIST' 'ELSE' 'WITH
        KW,←'ENDWITH' 'HOLD' 'ENDHOLD' 'TRAP' 'ENDTRAP' 'GOTO' 'RETURN' 'CONTINUE
          KW,←'SECTION' 'ENDSECTION' 'DISPOSABLE' 'ENDDISPOSABLE'
          KW,⍤¨⍨←':'
```

```
          msk←~KWϵ⍤⊂kws←n≠⍤⊂km←t=K
          ∨≠msk:('UNRECOGNIZED KEYWORD ',kws⊃⍤⊃⍳msk)⎕SIGNAL 2
          msk←kwsϵ':NAMESPACE' ':ENDNAMESPACE'
          ∨≠msk∧km≠tm:'NAMESPACE SCRIPTS MUST APPEAR AT THE TOP LEVEL'⎕SIGNAL 2
          msk←kwsϵKW~':NAMESPACE' ':ENDNAMESPACE' ':SECTION' ':ENDSECTION'
        ∨≠msk←msk∧~km≠tm:{msg←2'STRUCTURED STATEMENTS MUST APPEAR WITHIN TRAD-FNS
                  msg SIGNAL ϵ{x+⍳end[⍵]-x←pos[⍵]}⍤⍳km⌿msk}⍬

⍝ Verify system variables are valid
          SYSV←,⍤⊂'Á' 'A' 'AI' 'AN' 'AV' 'AVU' 'BASE' 'CT' 'D' 'DCT' 'DIV' 'DM'
          SYSV,←,⍤⊂'DMX' 'EXCEPTION' 'FAVAIL' 'FNAMES' 'FNUMS' 'FR' 'IO' 'LC' 'LX
          SYSV,←,⍤⊂'ML' 'NNAMES' 'NNUMS' 'NSI' 'NULL' 'PATH' 'PP' 'PW' 'RL' 'RSI
        SYSV,←,⍤⊂'RTL' 'SD' 'SE' 'SI' 'SM' 'STACK' 'TC' 'THIS' 'TID' 'TNAME' 'TNUMS
          SYSV,←,⍤⊂'TPOOL' 'TRACE' 'TRAP' 'TS' 'USING' 'WA' 'WSID' 'WX' 'XSI'
       SYSF←,⍤⊂'ARBIN' 'ARBOUT' 'AT' 'C' 'CLASS' 'CLEAR' 'CMD' 'CONV' 'CR' 'CS' 'CSV
          SYSF,←,⍤⊂'CY' 'DF' 'DL' 'DQ' 'DR' 'DT' 'ED' 'EM' 'EN' 'EX' 'EXPORT'
         SYSF,←,⍤⊂'FAPPEND' 'FCHK' 'FCOPY' 'FCREATE' 'FDROP' 'FERASE' 'FFT' 'IFFT
        SYSF,←,⍤⊂'FHIST' 'FHOLD' 'FIX' 'FLIB' 'FMT' 'FPROPS' 'FRDAC' 'FRDCI' 'FREAD
          SYSF,←,⍤⊂'FRENAME' 'FREPLACE' 'FRESIZE' 'FSIZE' 'FSTAC' 'FSTIE' 'FTIE'
         SYSF,←,⍤⊂'FUNTIE' 'FX' 'INSTANCES' 'JSON' 'KL' 'LOAD' 'LOCK' 'MAP' 'MKDIR
        SYSF,←,⍤⊂'MONITOR' 'NA' 'NAPPEND' 'NC' 'NCOPY' 'NCREATE' 'NDELETE' 'NERASE
          SYSF,←,⍤⊂'NEW' 'NEXISTS' 'NGET' 'NINFO' 'NL' 'NLOCK' 'NMOVE' 'NPARTS'
          SYSF,←,⍤⊂'NPUT' 'NQ' 'NR' 'NREAD' 'NRENAME' 'NREPLACE' 'NRESIZE' 'NS'
          SYSF,←,⍤⊂'NSIZE' 'NTIE' 'NUNTIE' 'NXLATE' 'OFF' 'OR' 'PFKEY' 'PROFILE'
          SYSF,←,⍤⊂'REFS' 'SAVE' 'SH' 'SHADOW' 'SIGNAL' 'SIZE' 'SR' 'SRC' 'STATE
        SYSF,←,⍤⊂'STOP' 'SVC' 'SVO' 'SVQ' 'SVR' 'SVS' 'TCNUMS' 'TGET' 'TKILL' 'TPUT
          SYSF,←,⍤⊂'TREQ' 'TSYNC' 'UCS' 'VR' 'VFI' 'WC' 'WG' 'WN' 'WS' 'XML' 'XT
          SYSD←,⍤⊂'OPT' 'R' 'S'
          ∨≠msk←(t=S)∧~nϵ'⎕',⍤⊂SYSV,SYSF,SYSD:{
                  ERR←2'INVALID SYSTEM VARIABLE, FUNCTION, OR OPERATOR'
                  ERR SIGNALϵpos[⍵]{⍺+⍳⍵-⍺}⍤⍳end[⍵]
          }⍳msk

⍝ Compute parent vector from d
          p←D2P d

⍝ Compute nameclass of dfns
          k←2×tϵF ⋄ k[∪p≠⍤⊂(t=P)∧nϵ⊂'⍺⍺']←3 ⋄ k[∪p≠⍤⊂(t=P)∧nϵ⊂'⍵⍵']←4

⍝ We will often wrap a set of nodes as children under a Z node
          gz←{z←⍵↑⍤⊂-0≠≠⍵ ⋄ ks←¯1↓⍵
                  t[z]←Z ⋄ p[ks]←⊃z ⋄ pos[z]←pos[⊃⍵] ⋄ end[z]←end[⊃⌽z,ks] ⋄ z}

⍝ Nest top-level root lines as Z nodes
          _←(gz 1⌽⊢)⍤⍳(t[i]=Z)⊂i←⍳d=0
          'Non-Z top-level node'assert t[⍳p=⍳≠p]=Z:
```

```
⍝ Nest all dfns expression bodies as Z nodes
        _←p[i]{end[⍺]←end[⊃⌽⍵] ⋄ gz¨⍵⊂⍨1,¯1↓t[⍵]=Z}⍤i←⍳⍨t[p]=F
        'Non-Z dfns body node'assert t[⍳⍨t[p]=F]=Z:

⍝ Drop/eliminate any Z nodes that are empty or blank
        _←p[i]{msk[⍺,⍵]←~∧⌿IN[pos[⍵]]∈WS}⍤i←⍳(t[p]=Z)∧p≠⍳≠p⍪msk←t≠Z
        tm n t k pos end(⌿⍨)←⊂msk ⋄ p←(⍳~msk)(⊢-1+⍳)msk⌿p

⍝ Parse Keyword structures
        nss←n∊⊂':NAMESPACE' ⋄ nse←n∊⊂':ENDNAMESPACE'
        ERR←':NAMESPACE KEYWORD MAY ONLY APPEAR AT BEGINNING OF A LINE'
        Z∨.≠t⌿⍨1⌽nss:ERR ⎕SIGNAL 2
        ERR←'NAMESPACE DECLARATION MAY HAVE ONLY A NAME OR BE EMPTY'
        ∨⌿(Z≠t⌿⍨¯1⌽nss)∧(V≠t⌿⍨¯1⌽nss)∨Z≠t⌿⍨¯2⌽nss:ERR ⎕SIGNAL 2
        ERR←':ENDNAMESPACE KEYWORD MUST APPEAR ALONE ON A LINE'
        ∨⌿Z≠t⌿⍨⊃1 ¯1∨.⌽⊂nse:ERR ⎕SIGNAL 2
        t[nsi←⍳1⌽nss]←M ⋄ t[nei←⍳1⌽nse]←-M
        n[i]←n[1+i←⍳(t=M)∧V=1⌽t] ⋄ end[nsi]←end[nei]
        x←⍳p=⍳≠p ⋄ d←+⍀(t[x]=M)+-t[x]=-M
        0≠⊃⌽d:':NAMESPACE KEYWORD MISSING :ENDNAMESPACE PAIR'⎕SIGNAL 2
        p[x]←x[D2P ¯1⌽d]

⍝ Delete unnecessary namespace nodes from the tree, leave only M's
        msk←~nss∨((¯1⌽nss)∧t=V)∨nse∨1⌽nse
        t k n pos end⌿⍨←⊂msk ⋄ p←(⍳~msk)(⊢-1+⍳)msk⌿p

⍝ PARSE LABELS ∘∘∘

⍝ Map guard statements to (G (Z ...) (Z ...))
        _←p[i]{
                0=+⌿m←':'=IN[pos[⍵]]:⍬
                ⊃m:'EMPTY GUARD TEST EXPRESSION'⎕SIGNAL 2
                1<+⌿m:'TOO MANY GUARDS'⎕SIGNAL 2
                t[⍺]←G ⋄ p[ti←gz⊃tx cq←2↑(⊂⍬)⌿⍨⍵⊂⍨1,¯1↓m]←⍺ ⋄ k[ti]←1
                ci←≠p ⋄ p,←⍺ ⋄ t k pos end⌿⍨←0 ⋄ n,←⊂'' ⋄ k[gz cq,ci]←1
        0}⍤i←⍳⍨t[p[p]]=F

⍝ Parse brackets and parentheses into ¯1 and Z nodes
        _←p[i]{
              x←IN[pos[⍵]] ⋄ bd←+⍀bm←(bo←'['=x)+-bc←']'=x ⋄ pd←+⍀pm←(po←'('=x)+
pc←')'=x
                           0≠⊃⌽bd:2'UNBALANCED  BRACKETS'SIGNAL  pos[⍵]{x+⍳(⌈⌿⍵)
x←⌊⌿⍺}⍤{⍵⌿⍨0≠bd}end[⍵]
                         0≠⊃⌽pd:2'UNBALANCED  PARENTHESES'SIGNAL  pos[⍵]{x+⍳(⌈⌿⍵)
x←⌊⌿⍺}⍤{⍵⌿⍨0≠pd}end[⍵]
```

```
                (po≠bd)∨.≠⊖pc≠bd:'OVERLAPPING BRACKETS AND PARENTHESES'⎕SIGNAL
                 p[⍵]←(⍺,⍵)[1+⍳¯1@{⍵=⍸⍵≠⍵}⌸2⌿+⍀¯1⊖bm+pm] ⋄ t[bo≠⍵]←¯1 ⋄ t[po≠⍵]←
                 end[po≠⍵]←end[⊖pc≠⍵] ⋄ end[bo≠⍵]←end[⊖bc≠⍵]
          0}⌸i←⍸(t[p]=Z)∧p≠⍳≠p
          t k n pos end≠⍤¨←⊂msk←~IN[pos]∊')' ⋄ p←(⍸~msk)(⊢-1+⍸)msk≠p

⍝ Convert semi-colon indexing into Z nodes in the ¯1 nodes
        _←p[i]{k[z←⊃⍪≠gz⌸g←⍵⊂⍤¯1⊖IN[pos[⍵]]∊';']←1 ⋄ t[z]←Z P[1=≢⌸g]}⌸i←⍸t[p]=¯

⍝ Mark bindable nodes
          bm←(t=V)∨(t=A)∧n∊,⍤'⎕⍞'
          bm←{bm⍪p[i]{bm[⍺]←(V ¯1≡t[⍵])∨∧≠bm[⍵]}⌸i←⍸(~bm[p])∧t[p]=Z}⍣≡bm

⍝ Binding nodes
          _←p[i]{
                   t[⍵≠⍤⊃(n[⍵]∊⊂,'←')∧0,¯1↓bm[⍵]]←B
                   b ∨←{(⊃⍤⌷x)(1↓⍤⌷x←⍵≠⍤⊃{t[⊃⍵]=B}⍤⌷⍵)}⍣¯1⊖⍤⌷⍵⊂⍤⊃1,¯1↓t[⍵]∊P B
                   ∨≠~bm[∊∨]:'CANNOT BIND ASSIGNMENT VALUE'⎕SIGNAL 2
                   p[⍵]←(⍺,b)[0,¯1↓+⍀t[⍵]=B]
                   n[b]←n[∊∨] ⋄ t[∊∨]←¯7 ⋄ pos[b]←pos[∊∨] ⋄ end[b]←end[⊃⊖⍵]
          0}⌸i←⍸(t[p]=Z)∧p≠⍳≠p
          t k n pos end≠⍤¨←⊂msk←t≠¯7 ⋄ p←(⍸~msk)(⊢-1+⍸)msk≠p

⍝ Mark unambiguous primitive kinds
        k[⍸(t=S)∧n∊'⎕',⍤'SYSV']←1 ⋄ k[⍸(t=S)∧n∊'⍞',⍤'SYSF']←2 ⋄ k[⍸(t=S)∧n∊'⎕',⍤'SYSD']←
          t[⍸t=S]←P
          k[⍸t∊A C N]←1 ⋄ k[⍸n∊,⍤'prmfs']←2 ⋄ k[⍸n∊,⍤'prmmo']←3 ⋄ k[⍸n∊,⍤'prmdo']←4
          k[⍸n∊,⍤'prmfo']←5
          k[i←⍸msk←(n∊⊂,'∘')∧1⊖n∊⊂,'.']←3 ⋄ end[i]←end[i+1] ⋄ n[i]←⊂,'∘.'
          t k n pos end≠⍤¨←⊂msk←~¯1⊖msk ⋄ p←(⍸~msk)(⊢-1+⍸)msk≠p

⍝ Anchor variables to earliest binding in matching frame
          rf←¯1@{~t[⍵]∊F G M}p[rz←I@{~(t[⍵]=Z)∧(t[p[⍵]]∊F G M)∨p[⍵]=⍵}⍣≡⍤¨p]
          rf[i]←p[i←⍸t=G] ⋄ rz[i]←i ⋄ rf←rf I@{rz∊p[i]⊢∘⊃⌸i←⍸t[p]=G}rf
          mk←{⍺[⍵],⍪n[⍵]}
          fr←rf mk⊢⍤fb←fb[⍳⍤¨rf mk⊢⍤fb←fb I∘(⍳⍤¨)U⍥rz mk⊢⍤fb←⍸t=B] ⋄ fb,←¯1
          vb←fb[fr⍳rf mk i]@(i←⍸t=V)⊢¯1⍴⍤¨≠p
          vb[i≠⍤¨(rz[i]<rz[b])∨(rz[i]=rz[b])∧i≥b←vb[i←i≠⍤¨vb[i]≠¯1]]←¯1
          _←{z/⍤¨¯1=vb[1⌷z]←fb[fr⍳⍥n I@1⊢z←rf I@0⊢⍵]}⍣≡⍥{rf[⍵],⍪⍵}⍸(t=V)∧vb=¯1
          ∨≠msk←(t=V)∧vb=¯1:{
             6'ALL VARIABLES MUST REFERENCE A BINDING'⎕SIGNAL∊pos[⍵]{⍺+⍳⍵-⍺}⍤end[⍵
          }⍸msk

⍝ ⍺/⍵ → V ; M → F0 ; ⍺⍺/⍵⍵ → P2
          t←V@(i←⍸(t=A)∧n∊,⍤'⍺⍵')⊢F@{t=M}t ⋄ vb[i]←i ⋄ k[⍸(t=P)∧n∊'⍺⍺' '⍵⍵']←2
```

```
⍝ Infer types of bindings, groups, and variables
          z x←↓⍉p[i]{⍺⍵}⌸i←⍳(t[p]∊B Z)∧p≠⍳≠p
          x←{⍵≠⍤¯1⊢∧⌿t[⍵]=¯1}U⍣¯1⊢x
         0∨.=≠⍤1⊢x:'BRACKET SYNTAX REQUIRES FUNCTION OR ARRAY TO ITS LEFT'⎕SIGNAL
          _←{
                   k[msk/z]←k[x/⍤1msk←(k[⊃⍤1⊢x]≠0)∧1=≠⍤1⊢x] ⋄ z x/⍤1←⊂⍤1msk
                   k[z/⍤1msk←k[⊃⍤1⊢x]=4]←3 ⋄ z x/⍤1←⊂⍤1msk
            k[z/⍤1msk←{(2 3 5∊⍤1k[⊃⍵])∨4=(⍵,≠k)[0⍳⍤1∧⌿k[⍵]=1]⌷k,0}∘⍣¯1⊢x]←2 ⋄ z x/⍤1←⊂⍤1
                   k[z/⍤1msk←k[⊃∘⍣¯1⊢x]=1]←1 ⋄ z x/⍤1←⊂⍤1msk
                   k[i]←k[⍸b[i←⍳⍤1t=V]]
          ≠z}⍣(=∨0=⊣)≠z
          'FAILED TO INFER ALL BINDING TYPES'assert 0=≠z:

⍝ Strand arrays into atoms
          i←⍸i⊣km←0<i←i[⍋⍋(i,⍤1⊢-∪p[i]),p[i←⍳⍤1t[p]∊B Z]]
          msk←(t[i]∊C N)∨msk∧⊃1 ¯1∨.∘⊂msk←km∧(t[i]∊A C N V Z)∧k[i]=1
          np←(≠p)+⍳≠ai←i/⍤1am←2>/msk⌿0 ⋄ p←(np@ai⍳≠p)[p] ⋄ p,←ai ⋄ km←2</0⌿msk
          t k n pos end(⊣,I)←⊂ai ⋄ k[ai]←1 6[∨/⍤1msk⊆t[i]≠N]
          t n pos(⊣@ai⍤1)←A(⊂'')(pos[km/i]) ⋄ p[msk/i]←ai[(msk←msk∧~am)/⍤¯1++⍀km]
          i←⍳(t[p]=A)∧(k[p]=6)∧t=N
          p,←i ⋄ t k n pos end(⊣,I)←⊂i ⋄ t k n(⊣@i⍤1)←A 1(⊂'')

⍝ PARSE B←D...
⍝ PARSE B←...D

⍝ Rationalize F[X]
          _←p[i]{
                   ⊃m←t[⍵]=¯1:'SYNTAX ERROR:NOTHING TO INDEX'⎕SIGNAL 2
                   k[⍵/⍤1m∧¯1∘⊃(k[⍵]∊2 3 5)∨¯1∘⊃k[⍵]=4]←4
          0}⌸i←⍳(t[p]∊B Z)∧(p≠⍳≠p)∧k[p]∊1 2
          i←⍳(t=¯1)∧k=4 ⋄ j←⍳(t[p]=¯1)∧k[p]=4
          (≠i)≠≠j:{
             2'AXIS REQUIRES SINGLE AXIS EXPRESSION'SIGNAL ∊pos[⍵]+⍳¨end[⍵]-pos[⍵
          }⊃⌿/{⊂⍺/⍤1⍵<≠⍵}⌸p[j]
          ∨/⍤1msk←t[j]≠Z:{
               2'AXIS REQUIRES NON-EMPTY AXIS EXPRESSION'SIGNAL ∊pos[⍵]+⍳¨end[⍵]
pos[⍵]
          }msk/p[j]
          p[j]←p[i] ⋄ t[i]←P ⋄ end[i]←1+pos[i]

⍝ Group function and value expressions
          i km←⌿/p[i]{(⍺⌿⍵)(0,1∨⍵)}⌸i←⍳(t[p]∊B Z)∧(p≠⍳≠p)∧k[p]∊1 2

⍝ Mask and verify dyadic operator right operands
          (dm←¯1∘⊃(k[i]=4)∧t[i]∊F P V Z)∨.∧(~km)∨k[i]∊0 3 4:{
                   'MISSING RIGHT OPERAND'⎕SIGNAL 2
```

```
          }⍬

  ⍝ Refine schizophrenic types
          k[i/⍨(k[i]=5)∧dm∨¯1⌽(~km)∨(~dm)∧k[i]∈1 6]←2 ⋄ k[i/⍨k[i]=5]←3

  ⍝ Rationalize ∘.
          jm←(t[i]=P)∧n[i]∈⊂,'∘.'
          jm∨.∧1⌽(~km)∨k[i]∈3 4:'MISSING OPERAND TO ∘.'⎕SIGNAL 2
        p←((ji←jm/i)@(jj←i/⍨¯1⌽jm)⍳≠p)[p] ⋄ t[ji,jj]←t[jj,ji] ⋄ k[ji,jj]←k[jj,ji
          n[ji,jj]←n[jj,ji] ⋄ pos[ji,jj]←pos[ji,ji] ⋄ end[ji,jj]←end[jj,jj]

  ⍝ Mask and verify monadic and dyadic operator left operands
          v/msk←(dm∧¯2⌽~km)∨(¯1⌽~km)∧mm←(k[i]=3)∧t[i]∈F P V Z:{
                  2'MISSING LEFT OPERAND'SIGNAL ∈pos[⍵]+⍳⍨end[⍵]-pos[⍵]
          }i/⍨msk
          msk←dm∨mm

  ⍝ Parse function expressions
          np←(≠p)+⍳xc←≠oi←msk/i ⋄ p←(np@oi⍳≠p)[p] ⋄ p,←oi ⋄ t k n pos end(⍪,I)←⊂o
          p[g/i]←oi[(g←(~msk)∧(1⌽msk)∨2⌽dm)/xc-⌽+⍀⌽msk]
          p[g/oi]←(g←msk/(1⌽mm)∨2⌽dm)/1⌽oi ⋄ t[oi]←O ⋄ n[oi]←⊂''
          pos[oi]←pos[g/i][msk/¯1++⍀g←(~msk)∧(1⌽mm)∨2⌽dm]
          ol←1+(k[i/⍨(2⌽mm)∨3⌽dm]=4)∨k[i/⍨(1⌽mm)∨2⌽dm]∈2 3
          or←(msk/dm)⍀1+k[dm/i]=2
          k[oi]←3 3⊥⍉or ol

  ⍝ Wrap all assignment values as Z nodes
          i km←⍪⌿/p[i]{(α⍪ω)(0,1∨ω)}⌸i←⍸(t[p]∈B Z)∧(p≠⍳≠p)∧k[p]∈1
          j←i/⍨msk←(t[i]=P)∧n[i]∈⊂,'←' ⋄ nz←(≠p)+⍳zc←+/msk
          p,←nz ⋄ t k n,←zc⍴¨Z 1(⊂'') ⋄ pos,←1+pos[j] ⋄ end,←end[p[j]]
          zm←¯1⌽msk ⋄ p[km/i]←(zpm/(i×~km)+zm⍀nz)[km/¯1++⍀zpm←zm∨~km]

  ⍝ This is the definition of a function value at this point
          isfn←{(t[⍵]∈O F)∨(t[⍵]∈B P V Z)∧k[⍵]=2}

  ⍝ Parse modified assignment to E4(V, F, Z)
          j←i/⍨m←msk∧(¯1⌽isfn i)∧¯2⌽(t[i]=V)∧k[i]=1 ⋄ p[zi←nz/⍨msk/m]←j
        p[i/⍨(1⌽m)∨2⌽m]←2/j ⋄ t k(⍱@j⍨)←E 4 ⋄ pos end n{α[⍵]@j⊢α}←vi zi,⊂vi←i/⍨2⌽

  ⍝ Parse bracket modified assignment to E4(E6, O2(F, P3(←)), Z)
          j←i/⍨m←msk∧(¯1⌽isfn i)∧(¯2⌽t[i]=¯1)∧¯3⌽(t[i]=V)∧k[i]=1
          p[zi←nz/⍨msk/m]←ei←i/⍨3⌽m ⋄ t k end(⍱@ei⍨)←E 4(end[zi])
          p t k n(⍱@(i/⍨2⌽m)⍨)←ei E 6(⊂'')
          p,←j ⋄ t,←Pρ⍨/j ⋄ k,←3ρ⍨/j ⋄ n,←(/j)ρ⊂,'←' ⋄ pos,←pos[j] ⋄ end,←end[j
          p t k n pos(⍱@j⍨)←ei O 2(⊂'')(pos[fi←i/⍨1⌽m]) ⋄ p[fi]←j
```

```
⍝ Parse bracket assignment to E4(E6, P2(←), Z)
          j←i⌿⍨m←msk∧(¯1⌽t[i]=¯1)∧¯2⌽(t[i]=V)∧k[i]=1 ⋄ p[zi←nz⌿⍨msk⌿m]←ei←i⌿⍨2⌽m
          t k end(¬⍀ei⍨)←E 4(end[zi]) ⋄ p t k n(¬⍀(i⌿⍨1⌽m)⍨)←ei E 6(⊂'')
          p t k(¬⍀j⍨)←ei P 2

⍝ Parse modified strand assignment
⍝ Parse strand assignment

⍝ SELECTIVE MODIFIED ASSIGNMENT
⍝ SELECTIVE ASSIGNMENT

⍝ Enclose V[X;...] for expression parsing
          i←i[⍋p[i←⍸(t[p]∊B Z)∧(k[p]=1)∧p≠⍳≠p]] ⋄ j←i⌿⍨jm←t[i]=¯1
          t[j]←A ⋄ k[j]←¯1 ⋄ p[i⌿⍨1⌽jm]←j

⍝ TRAINS

⍝ Parse expression sequences
          i km←⍪⌿p[i]{(⍺⍪⍵)(0,(2≤≠⍵)∧1∨⍵)}⍤⊃i←⍸(t[p]∊B Z)∧(k[p]=1)∧p≠⍳≠p
          msk←m2∨fm∧~¯1⌽m2←km∧(1⌽km)∧~fm←(t[i]=0)∨(t[i]≠A)∧k[i]=2
          t,←Eρ⍨xc←+⌿msk ⋄ k,←msk⌿msk+m2 ⋄ n,←xcρ⊂''
          pos,←pos[msk⌿i] ⋄ end,←end[p[msk⌿i]]
          p,←msk⌿¯1⌽(i×~km)+km×x←¯1+(≠p)++\msk ⋄ p[km⌿i]←km⌿x

⍝ Rationalize V[X;...]
          i←i[⍋p[i←⍸(t[p]=A)∧k[p]=¯1]] ⋄ msk←~2≠⌿¯1,ip←p[i] ⋄ ip←∪ip ⋄ nc←2×≠ip
          t[ip]←E ⋄ k[ip]←2 ⋄ n[ip]←⊂'' ⋄ p[msk⌿i]←msk⌿(≠p)+1+2×¯1++\~msk
          p,←2⌿ip ⋄ t,←ncρP E ⋄ k,←ncρ2 6 ⋄ n,←ncρ,⍨''[' '']
          pos,←2⌿pos[ip] ⋄ end,←∊(1+pos[ip]),⍪end[ip] ⋄ pos[ip]←pos[i⌿⍨~msk]

⍝ Sanity check
          ERR←'INVARIANT ERROR: Z node with multiple children'
          ERR assert(+⌿(t[p]=Z)∧p≠⍳≠p)=+⌿t=Z:

⍝ Count parentheses in source information
          ip←p[i←⍸(t[p]=Z)∧n[p]∊⊂,'('] ⋄ pos[i]←pos[ip] ⋄ end[i]←end[ip]

⍝ VERIFY Z/B NODE TYPES MATCH ACTUAL TYPE

⍝ Eliminate Z nodes from the tree
          zi←p I⍀{t[p[⍵]]=Z}⍤⍸⍤≡ki←⍸msk←(t[p]=Z)∧t≠Z
          p←(zi@ki⍳≠p)[p] ⋄ t k n pos end(¬⍀zi⍨)←t k n pos end I¨⊂ki
          t k n pos end⌿⍨←⊂msk←~msk∨t=Z ⋄ p←(⍸~msk)(⊢-1+⍸)msk⌿p

⍝ Compute Exports
          msk←(t=B)∧k[I⍀{t[⍵]≠F}⍤⍸⍤≡⍨p]=0
```

```
xn←(0ρ⊂''),msk⌿n ⋄ xt←msk⌿k

d i←P2D p ⋄ d n t k pos end I⍬⊢←⊂i ⋄ sym←∪''(,'ω')(,'α')'αα' 'ωω',n
(d t k(-symɩn)pos end)(xn xt)sym IN}
```

This code is used in chunk 4.
Uses TEST 12.

5.2 Compiler Transformations

23 ⟨*Compiler* 23⟩≡

```
TT←{((d t k n ss se)exp sym src)←ω ⋄ I←{(⊂ω)⌷α}
        A B C E F G K L M N O P S V Z←1+⍳15

 ⍝ Compute parent vector and reference scope
        r←I@{t[ω]≠F}⍤≡⍨p⍪2{p[ω]←α[α⍳ω]}⌿⊢∘⊂⌸d⍪p←⍳≠d

 ⍝ Lift Functions to top-level
        p,←n[i]←(≠p)+⍳≠i←⍸(t=F)∧p≠⍳≠p ⋄ t k n r(⍪,I)←⊂i ⋄ p r I⍨←⊂n[i]@i⊢⍳≠p
        t[i]←C

 ⍝ Wrap expressions as binding or return statements
        i←(⍸(~t∈F G)∧t[p]=F),{ω≠⍨2|⍳≠ω}⍸t[p]=G ⋄ p t k n r≠⍨←⊂m←2@i⊢1⍴⍨≠p
        p r i I⍨←⊂j←(+⍀m)-1 ⋄ n←j I@(0≤⊢)n ⋄ p[i]←j←i-1
        k[j]←-(k[r[j]]=0)∨0@({⊃⍵⌸p[j]})⊢(t[j]=B)∨(t[j]=E)∧k[j]=4 ⋄ t[j]←E

 ⍝ Lift guard tests
        p[i]←p[x←¯1+i←{ω≠⍨2|⍳≠ω}⍸t[p]=G] ⋄ t[i,x]←t[x,i] ⋄ k[i,x]←k[x,i]
        n[x]←n[i] ⋄ p←((x,i)@(i,x)⊢⍳≠p)[p]

 ⍝ Count strand and indexing children
        n[⍸(t∈A E)∧k=6]←0 ⋄ n[p≠⍨(t[p]∈A E)∧k[p]=6]+←1

 ⍝ Lift and flatten expressions
        p[i]←p[x←p I@{~t[p[ω]]∈F G}⍤≡⍨i←⍸t∈G A B C E O P V] ⋄ j←(⍵⍳i)[⍋⍵⍳x]
        p t k n r{α[ω]@i⊢α}←⊂j ⋄ p←(i@j⊢⍳≠p)[p]

 ⍝ Compute slots for each frame
        s←¯1,⍨∈⍳¨n[∪x]←⊢∘≠⌸x←0⌷⍉⍵e←∪I∘⍋⍨rn←r[b],⍪n[b←⍸t=B]

 ⍝ Compute frame depths
        d←(≠p)↑d ⋄ d[i←⍸t=F]←0 ⋄ _←{z⍪d[i]+←ω≠z←r[ω]}⍤≡⍨i ⋄ f←d[0⌷⍉⍵e],¯1

 ⍝ Record exported top-level bindings
        xi←⍸(t=B)∧k[r]=0

        p t k n f s r d xi sym}
```

This code is used in chunk 4.
Uses `src` 40a.

5.3 Code Generator

24 ⟨*Code Generator* 24⟩≡

```
GC←{
      p t k n fr sl rf fd xi sym←⍵ ⋄ A B C E F G K L M N O P S V Z←1+⍳15
      I←{(⊂⍵)⌷⍺} ⋄ com←{⊃{⍺,',',⍵}/⍵}
      ks←{⍵⊂[0]⍤(⊃⍵)=⍵[;0]} ⋄ nam←{'Δ'⎕R'__'⍠⍕⍤sym[⌊⍵]}

    syms ←,⍤'+'       '-'       '×'       '÷'       '*'       '⍟'
    nams ←     'add'   'sub'  'mul' 'div' 'exp' 'log' 'res'   'cir'   'min'  'max
    syms,←,⍤'<'       '≤'       '='       '≥'       '>'       '≠'       '
    nams,←     'lth'   'lte'  'eql' 'gte' 'gth' 'neq' 'not'   'and'   'lor'  'nan
    syms,←,⍤'⌷'          '['       '⍳'       '⍴'       ','       '⍪'
    nams,←     'sqd'   'brk'  'iot' 'rho' 'cat' 'ctf' 'rot'   'trn'   'rtf'  'mem
    syms,←,⍤'≡'          '≢'       '⊢'       '⊣'       '⍀'       '⊥'
    nams,←     'eqv'   'nqv'  'rgt' 'lft' 'enc' 'dec' 'red'   'rdf'   'scn'  'scf
    syms,←,⍤'↑'          '↓'       '⍤'       '⍨'       '.'       '⍤'
    nams,←     'tke'   'drp'  'map' 'com' 'dot' 'rnk' 'pow'   'jot'   'unq'  'int
    syms,←,⍤'⍋'          '⍒'       '∘.'      '⍷'       '⊂'       '⌹'       '⌷FFT
    nams,←     'gdu'   'gdd'  'oup' 'fnd' 'par' 'mdv' 'fft'   'ift'   'scl'  'nst
    syms,←,⍤'∇'          ';'       'α'       'ω'       'αα'      'ωω'
     nams,←      'this' 'span' 'l'            'r'          'aa'    'ww'    '

      gck← (A 1)(A 6)
      gcv← 'Aa' 'As'
      gck,←(B 1)(B 2)(B 3)(B 4)
      gcv,←'Bv' 'Bf' 'Bo' 'Bo'
      gck,←(C 1)(C 2)
      gcv,←'Ca' 'Cf'
      gck,←(E ¯2)(E ¯1)(E 0)(E 1)(E 2)(E 4)(E 6)
      gcv,←'Ec'      'Ek'    'Er' 'Em' 'Ed' 'Eb' 'Ei'
      gck,←(F 0)(F 2)(F 3)(F 4)
      gcv,←'Fz' 'Fn' 'Fm' 'Fd'
      gck,←(G 0)(N 1)
      gcv,←'Gd' 'Na'
      gck,←(O 1)(O 2)(O 4) (O 5) (O 7) (O 8)
      gcv,←'Ov' 'Of' 'Ovv' 'Ofv' 'Ovf' 'Off'
      gck,←(P 0)(P 1)(P 2)(P 3)(P 4)
      gcv,←'Pv' 'Pv' 'Pf' 'Po' 'Po'
      gck,←(V 0)(V 1)(V 2)(V 3)(V 4)
      gcv,←'Va' 'Va' 'Vf' 'Vo' 'Vo'
      gcv,←⊂'{''/* Unhandled '',(⍕⍺),'' */'',NL}'
      NL←⎕UCS 13 10

      pref ←⊂'#include "codfns.h"'
      pref,←⊂''
```

```
        pref,←⊂'EXPORT int'
        pref,←⊂'DyalogGetInterpreterFunctions(void *p)'
        pref,←⊂'{'
        pref,←⊂'    return set_dwafns(p);'
        pref,←⊂'}'
        pref,←⊂''

        Bf←{id←sym⊃⍵⍵|4⊃⍺
                z ←⊂id,' = retain_cell(stkhd[-1]);'
        z}

        Cf←{id←⍕4⊃⍺
                z ←⊂'mk_closure((struct closure **)stkhd++, fn',id,', 0);'
        z}

        Ek←{
                z ←⊂'release_cell(*--stkhd);'
                z,←⊂''
        z}

        Em←{
                z ←⊂'c = *--stkhd;'
                z,←⊂'w = *--stkhd;'
                z,←⊂'(c->fn)((struct array **)stkhd++, NULL, w, c->fv);'
                z,←⊂'release_cell(c);'
                z,←⊂'release_cell(w);'
        z}

        Er←{
                z ←⊂'*z = *--stkhd;'
                z,←⊂'goto cleanup;'
                z,←⊂''
        z}

        Fn←{id←⍕5⊃⍺ ⋄ x←⍵⊃⍨⍸≠⍵ ⋄ t←2⌷x ⋄ k←3⌷x
                hsw←(t=0)∨(t=E)∧k∈1 2 ⋄ hsa←((t=E)∧k=2)∨(t=0)∧k∈4 5 7 8
                z ←⊂'int'
                   z,←⊂'fn',id,'(struct array **z, struct array *l, struct ar
ray *r, void *fv[])'
                z,←⊂'{'
                z,←⊂'       void    *stk[128];'
                z,←⊂'       void    **stkhd;'
                z,←hsw/⊂' void    *w;'
                z,←hsa/⊂' void    *a;'
                z,←hsw/⊂' struct  closure *c;'
                z,←⊂''
```

```
                z,←⊂'       stkhd = &stk[0];'
                z,←⊂''
                z,← ' ',¨⊃,≠dis¨⍵
                z,←⊂'       *z = NULL;'
                z,←⊂''
                z,←⊂'cleanup:'
                z,←⊂'       return 0;'
                z,←⊂'}'
                z,←⊂''
        z}

        Fz←{id←⍕5⊃⍺ ⋄ awc←∨≠(3⌷x){(⍵∈A O)∨(⍵=E)∧⍺>0}2⌷x←⎕⊃¯,≠⍵
                z ←⊂'int init',id,' = 0;'
                z,←⊂''
                z,←⊂'EXPORT int'
                z,←⊂'init(void)'
                z,←⊂'{'
                z,←⊂' return fn',id,'(NULL, NULL, NULL, NULL);'
                z,←⊂'}'
                z,←⊂''
                z,←⊂'int'
                   z,←⊂'fn',id,'(struct array **z, struct array *l, struct ar
ray *r, void *fv[])'
                z,←⊂'{'
                z,←⊂'       void    *stk[128];'
                z,←⊂'       void    **stkhd;'
                z,← awc≠⊂'         void    *a, *w;'
                z,← awc≠⊂'         struct  closure *c;'
                z,←⊂''
                z,←⊂'       if (init',id,')'
                z,←⊂'               return 0;'
                z,←⊂''
                z,←⊂'       stkhd = &stk[0];'
                z,←⊂'       init',id,' = 1;'
                z,←⊂'       cdf_init();'
                z,←⊂''
                z,← ' ',¨⊃,≠dis¨⍵
                z,←⊂'       return 0;'
                z,←⊂'}'
                z,←⊂''
        z}

        Pf←{id←(symsɩsym[|4⊃⍺])⊃nams
                z ←⊂'*stkhd++ = retain_cell(',id,');'
        z}
```

```
        Va←{id←(|4⊃⍺)⊃'' 'r' 'l' 'aa' 'ww',5↓sym
                z ←⊂'*stkhd++ = retain_cell(',id,');'
        z}

        Zp←{n←'fn',⍕⍵
                k[⍵]∊0 2:{
                        z ←⊂'int'
                          z,←⊂n,'(struct array **z, struct array *l, struct ar
ray *r, void *fv[]);'
                        z,←⊂''
                z}⍵
                'UNKNOWN FUNCTION TYPE'⎕SIGNAL 16
        }

        Zx←{n←sym⊃⍨⍸|n[⍵] ⋄ rid←⍕rf[⍵]
                k[⍵]=0:⊂''
                k[⍵]=1:{
                        z ←⊂'struct array *',n,';'
                z}⍵
                k[⍵]=2:{
                        z ←⊂'struct closure *',n,';'
                        z,←⊂''
                        z,←⊂'EXPORT int'
                    z,←⊂n,'_dwa(struct localp *zp, struct localp *lp, struct lo
calp *rp)'
                        z,←⊂'{'
                        z,←⊂'       struct array *z, *l, *r;'
                        z,←⊂'       int err;'
                        z,←⊂''
                        z,←⊂'       l = NULL;'
                        z,←⊂'       r = NULL;'
                        z,←⊂''
                        z,←⊂'       fn',rid,'(NULL, NULL, NULL, NULL);'
                        z,←⊂''
                        z,←⊂'       err = 0;'
                        z,←⊂''
                        z,←⊂'       if (lp)'
                        z,←⊂'               err = dwa2array(&l, lp->pocket);'
                        z,←⊂''
                        z,←⊂'       if (err)'
                        z,←⊂'               dwa_error(err);;'
                        z,←⊂''
                        z,←⊂'       if (rp)'
                        z,←⊂'               dwa2array(&r, rp->pocket);'
                        z,←⊂''
                        z,←⊂'       if (err) {'
```

```
                        z,←⊂'                release_array(l);'
                        z,←⊂'                dwa_error(err);'
                        z,←⊂'        }'
                        z,←⊂''
                        z,←⊂'        err = (',n,'->fn)(&z, l, r, ',n,'->fv);'
                        z,←⊂''
                        z,←⊂'        release_array(l);'
                        z,←⊂'        release_array(r);'
                        z,←⊂''
                        z,←⊂'        if (err)'
                        z,←⊂'                dwa_error(err);'
                        z,←⊂''
                        z,←⊂'        err = array2dwa(NULL, z, zp);'
                        z,←⊂'        release_array(z);'
                        z,←⊂''
                        z,←⊂'        if (err)'
                        z,←⊂'                dwa_error(err);'
                        z,←⊂''
                        z,←⊂'        return 0;'
                        z,←⊂'}'
                        z,←⊂''
                z}⍵
                ⍮'''UNKNOWN EXPORT TYPE''⎕SIGNAL 16'
        }

        d i←P2D p ⋄ ast←(⍉↑d p t k n(⍳≠p)fr sl fd)[i;]
        NOTFOUND←{('[GC] UNSUPPORTED NODE TYPE ',NΔ[⊃⍵],⍕⊃ϕ⍵)⎕SIGNAL 16}
    dis←{0=2⊃h←,1↑⍵:'' ⋄ (≠gck)=i←gck⍳⊂h[2 3]:NOTFOUND h[2 3] ⋄ h(⍮i⊃gcv)ks 1↓⍵
        z←ϵ,⍪NL¨pref,⊃,≠(,≠Zp¨⍸t=F),(,≠Zx¨xi),(⊂⊂''),dis¨ks ast
        z}
```

This code is used in chunk 4.
Uses codfns 4.

5.4 Backend C Compiler Interface

29 ⟨*Interface to the backend C compiler* 29⟩≡

```
CC←{
        vsbat←VSΔPATH,'\VC\Auxiliary\Build\vcvarsall.bat'
        tie←{0::⎕SIGNAL ⎕EN ⋄ 22::⍵ ⎕NCREATE 0 ⋄ 0 ⎕NRESIZE ⍵ ⎕NTIE 0}
      put←{s←(⍕128+256|128+'UTF-8'⎕UCS ⍺)⎕NAPPEND(t←tie ⍵)83 ⋄ 1:r←s⍪⎕NUNTIE t
        opsys←{⍵⊃⍨'Win' 'Lin' 'Mac'⍳⊂3↑⊃'.'⎕WG'APLVersion'}
        soext←{opsys'.dll' '.so' '.dylib'}
        ccf←{' -o ''',⍵,'.',⍺,''' ''',⍵,'.c'' -laf',AFΔLIB,' > ',⍵,'.log 2>&1'}
      cci←{'-I''',AFΔPREFIX,'/include'' -L''',AFΔPREFIX,opsys''' ' '/lib64'' ' '/
        cco←'-std=c99 -Ofast -g -Wall -fPIC -shared -Wno-parentheses '
        cco,←'-Wno-misleading-indentation '
        ucc←{⍵⍵(⎕SH ⍺⍺,' ',cco,cci,ccf)⍵}
        gcc←'gcc'ucc'so'
        clang←'clang'ucc'dylib'
        vsco←{z←'/W3 /wd4102 /wd4275 /O2 /Zc:inline /Zi /FS /Fd"',⍵,'.pdb" '
                z,←'/WX /MD /EHsc /nologo '
                z,'/I"%AF_PATH%\include" /D "NOMINMAX" /D "AF_DEBUG" '}
        vslo←{z←'/link /DLL /OPT:REF /INCREMENTAL:NO /SUBSYSTEM:WINDOWS '
            z,←'/LIBPATH:"%AF_PATH%\lib" /OPT:ICF /ERRORREPORT:PROMPT /TLBID:1
                z,'/DYNAMICBASE "af', AFΔLIB, '.lib" "codfns.lib" '}
        vsc0←{~⎕NEXISTS vsbat:'VISUAL C?'⎕SIGNAL 99 ⋄ '"',vsbat,'" amd64'}
        vsc1←{' && cd "',(⊃⎕CMD'echo %CD%'),'" && cl ',(vsco ⍵),' "',⍵,'.c" '}
        vsc2←{(vslo ⍵),'/OUT:"',⍵,'.dll" > "',⍵,'.log""'}
        vsc←{⎕CMD ('%comspec% /C ',vsc0,vsc1,vsc2)⍵}
        _←(⍎opsys'vsc' 'gcc' 'clang')⍺⍺⍵ put ⍺,'.c'⍪1 ⎕NDELETE f←⍺,soext⍬
        ⎕←⍪⊃⎕NGET(⍺,'.log')1
        ⎕NEXISTS f:f ⋄ 'COMPILE ERROR' ⎕SIGNAL 22}
```

This code is used in chunk 4.
Uses AFΔLIB 7, AFΔPREFIX 7, codfns 4, put 33b, tie 33b, vsbat 40b, and VSΔPATH 8.

5.5 Linking with Dyalog

30a ⟨*Linking with Dyalog* 30a⟩≡

```
NS←{
        MKA←{mka⊂⍵} ⋄ EXA←{exa ⍬ ⍵}
        Display←{⍺←'Co-dfns' ⋄ W←w_new⊂⍺ ⋄ 777::w_del W
                w_del W⊣W ⍺⍺{w_close ⍺:⍬⍴'⎕SIGNAL 777' ⋄ ⍺ ⍺⍺ ⍵}⍣⍵⍵⊢⍵}
        LoadImage←{⍺←1 ⋄ ~⎕NEXISTS ⍵:⎕SIGNAL 22 ⋄ loadimg ⍬ ⍵ ⍺}
        SaveImage←{⍺←'image.png' ⋄ saveimg ⍵ ⍺}
        Image←{~2 3∨.=≢⍴⍵:⎕SIGNAL 4 ⋄ (3≠⊃⌽⍴⍵)∧3=≢⍴⍵:⎕SIGNAL 5 ⋄ ⍵⊣w_img ⍵ ⍺}
        Plot←{2≠≢≢⍴⍵:⎕SIGNAL 4 ⋄ ~2 3∨.=1⊃⌽⍴⍵:⎕SIGNAL 5 ⋄ ⍵⊣w_plot (⌽⍵) ⍺}
        Histogram←{⍵⊣w_hist ⍵,⍺}
        Rtm∆Init←{
                _←'w_new'    ⎕NA'P ',⍵,'|w_new           <C[]'
                _←'w_close'⎕NA'I ',⍵,'|w_close P'
                _←'w_del'    ⎕NA                    ⍵,'|w_del             P'
                _←'w_img'    ⎕NA                    ⍵,'|w_img             <PP P
                _←'w_plot' ⎕NA                      ⍵,'|w_plot    <PP P'
                _←'w_hist' ⎕NA                      ⍵,'|w_hist    <PP F8   F8 P
                _←'loadimg'⎕NA                      ⍵,'|loadimg >PP <C[] I'
                _←'saveimg'⎕NA                      ⍵,'|saveimg <PP <C[]'
                _←'exa'              ⎕NA                    ⍵,'|exarray >PP P'
                _←'mka'              ⎕NA'P ',⍵,'|mkarray <PP'
               _←'FREA'            ⎕NA                  ⍵,'|frea             P
                _←'Sync'             ⎕NA                   ⍵,'|cd_sync'
                0 0 ⍴ ⍬}
        mkna←{⍺,'|',('∆'⎕R'__'⊢⍵),'_cdf P P P'}
        mkf←{fn←⍺,'|',('∆'⎕R'__'⊢⍵),'_dwa ' ⋄ mon dya←⍵⍤¨'_mon' '_dya'
                z←('Z←{A}',⍵,' W')(':If 0=⎕NC''⍙.',mon,'''')
          z,←(mon dya{'''',⍺,'''⍙.⎕NA''',fn,⍵,' <PP'''}¨'>PP P' '>PP <PP'),⊂':E
          z,':If 0=⎕NC''A'''('Z←⍙.',mon,' 0 0 W')':Else'('Z←⍙.',dya,' 0 A W')':E
       ns←#.⎕NS⍬ ⋄ _←'∆⍙'ns.⎕NS¨⊂⍬ ⋄ ∆ ⍙←ns.(∆ ⍙) ⋄ ∆.names←(0⍴⊂''),(2=1⊃⍺)≠0⊃
     fns←'Rtm∆Init' 'MKA' 'EXA' 'Display' 'LoadImage' 'SaveImage' 'Image' 'Plot
       fns,←'Histogram' 'soext' 'opsys' 'mkna'
       _←∆.⎕FX⍤⎕CR¨fns ⋄ ∆.(decls←⍵⍤mkna¨names) ⋄ _←ns.⎕FX¨(⊂''),⍵⍤mkf¨∆.name
    _←∆.⎕FX'Z←Init'('Z←Rtm∆Init ''',⍵,''''')'⍎0≠⍤¨0=≢names' 'names ##.⍙.⎕NA¨dec
       ns}
```

This code is used in chunk 4.
Uses PP 33a.

6 Co-dfns Runtime

30b ⟨*Implementation of APL Primitives* 30b⟩≡

```
⍝ TBW
```

Root chunk (not used in this document).

31a ⟨*C Runtime Support* 31a⟩≡
```
/* TBW */
```
Root chunk (not used in this document).

31b ⟨*C Runtime Header* 31b⟩≡
```
/* TBW */
```
Root chunk (not used in this document).

7 Utilities

7.1 AST Pretty-printing

31c ⟨*Pretty-printing AST trees* 31c⟩≡
```
dct←{α[(2×2≠/n,0)+(1↑⍨≠m)+m+n←⍋⍒\⍋m←' '≠αα ⍵]⍵⍵ ⍵}
dlk←{((x⌷⍴⍵)↑[x←2|1+⍵⍵]α),[⍵⍵]αα@(⊂0 0)⍤('┌'=⊃⍵)⊢⍵}
dwh←{⍵('┬'dlk 1)' │├┌└─'(0⌷⍉)dct,⊃⍪/((≠¨α),¨⊂⌈/≠∘⍉¨α)↑¨α}
dwv←{⍵('├'dlk 0)' ─┬┌┐│'(0⌷⊢)dct(⊣⍪1↓⊢)⊃{α,' ',⍵}/(1+⌈/≠¨α){α↑⍵⍪⍨'│'↑⍨≠⍉⍵}¨α}

pp3←{α←'○' ⋄ d←(⍳≠⍵)≠⍵ ⋄ _←{z⊣d+←⍵≠z←α[⍵]}⍤≡⍨⍵ ⋄ lbl←α⍴⍨≠⍵
        lyr←{i←⍸⍨α=d ⋄ k v←↓⍉⍵⍵[i],∘⊂⌸i ⋄ (⍵∘{α[⍵]}¨v)αα¨@k⊢⍵}⍵
        (⍵=⍳≠⍵)≠⊃αα lyr≠(1+⍳⌈/d),⊂⍉∘⍪∘⍪¨lbl}

lb3←{α←⍳≠⊃⍵
        '(',¨')',¨⍨{α,';',⍵}≠⍪¨(NΔ{α[⍵]}@2⊢(2⊃⍵){α[⍸⍵]}@{0>⍵}@4↑⊃⍵)[α;]}
```
This code is used in chunk 4.
Defines:
`dct`, never used.
`dlk`, never used.
`dwh`, never used.
`dwv`, never used.
`lb3`, never used.
`pp3`, never used.

7.2 Debugging utilities

The following utilities help to improve quality of life when working with the Co-dfns source code.

The DISPLAY function is taken from https://dfns.dyalog.com and helps to make debugging easier by allowing us to thread DISPLAY calls into expressions. I prefer to do something like this:

```
... {ω⊣⎕←#.DISPLAY ω} ...
```

The function itself returns the character rendering of the code, so the above little expression is one that I use to insert and do debugging within an expression.

32 ⟨DISPLAY *Utility* 32⟩≡

```
DISPLAY←{⎕IO ⎕ML←0                                                    ⍝ Bo
play of array.

      α←1 ⋄ chars←α⊃'..''''|-' '┌┐└┘|─'                    ⍝ α: 0-clunky, 1-smooth

      tl tr bl br vt hz←chars                                        ⍝ Top left

      box←{
            vrt hrz←(¯1+⍴ω)⍴¨vt hz                                    ⍝ Vert. ar
             top←(hz,'⊖→')[¯1↑α],hrz                                  ⍝ Up
per border with axis.
            bot←(⊃α),hrz
der with type.
            rgt←tr,vt,vrt,br
            lax←(vt,'⍬↓')[¯1↓1↓α],¨⊂vrt                    ⍝ Left side(s) wi
            lft←↖tl,(↑lax),bl                                          ⍝
            lft,(top⍪ω⍪bot),rgt                                        ⍝
        }

      deco←{α←type open ω ⋄ α,axes ω}                    ⍝ Type and axes vector
      axes←{(-2⌈⍴⍴ω)↑1+×⍴ω}                                          ⍝ Ar
ray axis types.
      open←{(1⌈⍴ω)⍴⍴ω}
pose null axes.
      trim←{(~1 1⍷∧≠ω=' ')/ω}                                      ⍝ Re
move extra blank cols.
      type←{{(1=⍴ω)⊃'+'ω}∪,char¨ω}                       ⍝ Simple array type
        char←{⍬≡⍴ω:hz ⋄ (⊃ω∊'¯',⎕D)⊃'#~'}⍤⌶              ⍝ Simple scalar type.
       line←{(6≠10|⎕DR' 'ω)⊃' -'}                                    ⍝ un
derline for atom.

      {
cursively box arrays:
```

```
                0=≡⍵:' '⍪(open ⎕FMT ⍵)⍪line ⍵                    ⍝ Simple scalar
                1 ⍬≡(≡⍵)(⍴⍵):'∇' 0 0 box ⎕FMT ⍵                  ⍝ Object rep: ⎕OR
                   1=≡⍵:(deco ⍵)box open ⎕FMT open ⍵             ⍝ Simple array.
                   ('ϵ'deco ⍵)box trim ⎕FMT ∇¨open ⍵             ⍝ Nested array.
          }⍵
  }
```

Root chunk (not used in this document).
Defines:
  DISPLAY, used in chunks 33a and 35.
Uses ⎕IO 6a and ⎕ML 6a.

I also define a function PP that encapsulates the above usage pattern that I like to use, making the whole thing less verbose and a little more convenient.

33a ⟨PP *Utility* 33a⟩≡

```
PP←{⍵⊣⎕←#.DISPLAY ⍵}
```

Root chunk (not used in this document).
Defines:
  PP, used in chunks 30a and 35.
Uses DISPLAY 32.

7.3 Reading and Writing Files

It is helpful to be able to easily write files to disk, and the following put and tie utilities help us to do so when we want to. These are pretty standard, but they could maybe be replaced by ⎕NPUT or something like that.

33b ⟨*Basic* tie *and* put *utilities* 33b⟩≡

```
tie←{
        0::⎕SIGNAL ⎕EN
        22::⍵ ⎕NCREATE 0
        0 ⎕NRESIZE ⍵ ⎕NTIE 0
}

put←{
        s←(¯128+256|128+'UTF-8'⎕UCS ⍵)⎕NAPPEND(t←tie ⍺)83
        1:r←s⊣⎕NUNTIE t
}
```

This code is used in chunk 39c.
Defines:
  put, used in chunks 29, 39c, and 40a.
  tie, used in chunks 29 and 39c.

7.4 XML Rendering

34 ⟨*XML Rendering* 34⟩≡

```
Xml←{α←0 ⋄ ast←α{d i←P2D⊃ω ⋄ i⊃{ω[α]}¨(⊂d),1↓α↓ω}⍤(0≠α)⊢ω ⋄ d t k n←4↑ast
        cls←NΔ[t],¨('-..'[1+×k]),¨⍕¨|k ⋄ fld←{((≠ω)↑3↓fΔ),⍪ω}¨↓⍉↑3↓ast
        ⎕XML⍉↑d cls(⊂'')fld}
```

This code is used in chunk 4.

8 Developer Infrastructure

8.1 Building the Compiler

The Co-dfns compiler is written, developed, and distributed as a literate program. For more information about literate programming, see the resources available at `http://literateprogramming.com/`. We use noweb as our preferred literate programming tool because it is eminently simple, while still handling the majority of our needs and producing high quality output in LaTeX format with all the important elements of literate programming, including live hyperlinking and cross-references.

8.1.1 Tangling the Source

The process of tangling produces the executable source code for the compiler. Importantly, the tangled output is *not* meant to be used as the primary means of reading or debugging the source. Instead, it is meant primarily as the machine readable version of the code only. We intend `codfns.nw` to tangle into the following files based on specific chunks that have been provided in this document.

Chunk	File Output
⟨* 4⟩	`src\codfns.apln`
⟨*C Runtime Support* 31a⟩	`rtm\runtime.c`
⟨*C Runtime Header* 31b⟩	`rtm\codfns.h`
⟨*Implementation of APL Primitives* 30b⟩	`rtm\prim.apln`
⟨`DISPLAY` *Utility* 32⟩	`src\DISPLAY.aplf`
⟨`MK∆RTM` *Command* 39c⟩	`src\MK∆RTM.aplf`
⟨`PP` *Utility* 33a⟩	`src\PP.aplf`
⟨`TANGLE` *Command* 37⟩	`src\TANGLE.aplf`
⟨`TEST` *Function* 12⟩	`src\TEST.aplf`
⟨`WEAVE` *Command* 39a⟩	`src\WEAVE.aplf`
⟨*Tangle Script* 35⟩	`TANGLE.sh`
⟨*Weave Script* 38⟩	`WEAVE.sh`

The following bash script creates these files.

35 ⟨*Tangle Script* 35⟩≡

```
#!/bin/bash
echo "Tangling src/codfns.apln..."
notangle codfns.nw > src/codfns.apln

echo "Tangling src/DISPLAY.aplf..."
notangle -R'[[DISPLAY]] Utility' codfns.nw > src/DISPLAY.aplf
```

```
echo "Tangling src/MK∆RTM.aplf..."
notangle -R'[[MK∆RTM]] Command' codfns.nw > src/MK∆RTM.aplf

echo "Tangling src/PP.aplf..."
notangle -R'[[PP]] Utility' codfns.nw > src/PP.aplf

echo "Tangling src/TANGLE.aplf..."
notangle -R'[[TANGLE]] Command' codfns.nw > src/TANGLE.aplf

echo "Tangling src/TEST.aplf..."
notangle -R'[[TEST]] Function' codfns.nw > src/TEST.aplf

echo "Tangling src/WEAVE.aplf..."
notangle -R'[[WEAVE]] Command' codfns.nw > src/WEAVE.aplf

echo "Tangling TANGLE.sh..."
notangle -R'Tangle Script' codfns.nw > TANGLE.sh

echo "Tangling WEAVE.sh..."
notangle -R'Weave Script' codfns.nw > WEAVE.sh
```

Root chunk (not used in this document).
Uses codfns 4, DISPLAY 32, MK∆RTM 39c, PP 33a, src 40a, TANGLE 37, TEST 12, and WEAVE 39a.

On Windows, the best way that we have found to do this is via the Cygwin project. This document assumes that you have already successfully built and installed via Cygwin a working Icon-driven noweb installation.

Users who prefer to work in a UNIX fashion via Cygwin or some other subsystem on Windows can follow the build scripts directly. For developers who prefer to work in a primarily Windows environment, the following build scripts assist in handling the calls into Cygwin so that you do not need to have a Cygwin terminal open all the time.

When tangled to the `TANGLE.aplf` file, this script will enable the user simply type `TANGLE` to update the code tree from within a Dyalog APL Session. This is much more convenient than keeping a Cygwin Terminal session open along with a Dyalog APL session while programming. At the moment, this file is only designed to work on Windows with Cygwin, but we should in principle extend this to work on Linux and Mac OS X style machines as well.

Note: this command expects to be run from within the root of the repository, not from, say, within the `testing` *directory.*

37 ⟨`TANGLE` *Command* 37⟩≡

```
TANGLE;SH;CWD;CD
SH←'C:\cygwin64\bin\bash.exe -l -c '
CWD←⊃⎕CMD'C:\cygwin64\bin\bash.exe -c pwd'
CD←'cd ''',CWD,''''

⎕CMD SH,'"',CD,' && ./TANGLE.sh"'
```

Root chunk (not used in this document).
Defines:
`TANGLE`, used in chunk 35.

8.1.2 Weaving the Source

Weaving is the process by which we produce the final printed output of this document, intended for reading and general human consumption. We rely on the LaTeX typesetting system to do this. Moreover, because we make heavy use of UTF-8 and prefer to have our own fonts installed and used, it is necessary to use the `xelatex` system instead of the typical LaTeX engine. In order to get the indexing right, we must run the engine twice. The first run will update the indexing files that will be picked up on the second run and incorporated into the final document. Note, we have tried to use the `lualatex` engine, which in theory should work just as well as the `xelatex` engine, but we get a strange error relating to noweb's style file, so we stick with `xelatex` for now.

Running this script also depends on having the appropriate fonts installed. In this case, please ensure that the following fonts are installed in your Windows font system so that they can be picked up by the TeX engine.

- Libre Baskerville (Regular, Italic, Bold)
- APL385 Unicode
- Lucida Sans Unicode
- Cambria Math

If you do not wish to use these fonts, then see the top of the `codfns.nw` file and edit the font specifications to the fonts that you do wish to use.

Note the use of `-delay -index` for options. We want to generate indexing, but we also need to make sure that we can use some of our own packages in the system,

Note: this command expects to be run from within the root of the repository, not from, say, within the `testing` *directory.*

38 ⟨*Weave Script* 38⟩≡

```
#!/bin/bash
mkdir -p woven
noweave -delay -index codfns.nw > woven/codfns.tex
cd woven
xelatex codfns
xelatex codfns
```

Root chunk (not used in this document).
Uses `codfns` 4.

Like the ⟨TANGLE *Command* 37⟩, the following command, when tangled to the WEAVE.aplf file permits the Windows programmer to simply execute WEAVE in a the Dyalog APL session and weave the documents together.

39a ⟨WEAVE *Command* 39a⟩≡

```
WEAVE;SH;CWD;CD
SH←'C:\cygwin64\bin\mintty.exe -e C:\cygwin64\bin\bash.exe -l -c '
CWD←⊃⎕CMD'C:\cygwin64\bin\bash.exe -c pwd'
CD←'cd ''',CWD,''''

⎕CMD SH,'"',CD,' && ./WEAVE.sh"'
```

Root chunk (not used in this document).
Defines:
WEAVE, used in chunk 35.

8.2 Building the Runtime

One of our goals with the Co-dfns runtime is to write as much of it as possible in APL. This means that we want to have at minimum a very small kernel that has been written in C, while most of the rest of the code is implemented in some APL files. This leads to a three part breakdown of the process to build the runtime.

39b ⟨*Build the runtime* 39b⟩≡

```
⟨Compile the primitives in prim.apln 40a⟩
⟨Build codfns.dll DLL 40b⟩
⟨Copy the runtime files into tests\ 41⟩
```

This code is used in chunk 39c.

We define the command MK∆RTM to build the runtime. This command takes a path to the root directory of the Co-dfns repository; this is to allow us to rebuild the runtime from anywhere in the system if we so choose.

39c ⟨MK∆RTM *Command* 39c⟩≡

```
MK∆RTM path;prim;put;src;tie;vsbat;vsc;wsd

⟨Basic tie and put utilities 33b⟩
⟨Build the runtime 39b⟩
```

Root chunk (not used in this document).
Defines:
MK∆RTM, used in chunk 35.
Uses put 33b, src 40a, tie 33b, vsbat 40b, and wsd 40b.

The first step we must take is producing an appropriate C file that contains the primitives that we have defined in `prim.apln`. This means that we want to only compile the code in `prim.apln` as far as producing the C code. Since we do not have a full blown runtime yet, we will be compiling the `prim.c` file along with the rest of the runtime code, instead of the normal build process, which assumes that we already have a working runtime. This means that we only invoke the `GC TT PS` passes of the compiler pipeline, while avoiding the `CC` pass. We use the SALT system to load the source from `prim.apln` and then run the compiler passes that we want before storing the resulting code in the `rtm\prim.c` file.

40a ⟨*Compile the primitives in* `prim.apln` 40a⟩≡

```
src←⎕SRC ⎕SE.SALT.Load path,'\rtm\prim.apln'
(path,'\rtm\prim.c')put codfns.{GC TT PS ⍵}src
```

This code is used in chunk 39b.
Defines:
`src`, used in chunks 9, 23, 35, and 39c.
Uses `codfns` 4 and `put` 33b.

Once we have the `rtm\prim.c` file written appropriately, we can run the main compiler process. For simplicity, we just compile all of the `.c` files that are found in the `rtm\` subdirectory. We must ensure that we are appropriatelly invoking our ArrayFire dependencies as well as producing the appropriate debugging symbols most of the time.

40b ⟨*Build* `codfns.dll` *DLL* 40b⟩≡

```
vsbat←#.codfns.VS∆PATH,'\VC\Auxiliary\Build\vcvarsall.bat'
wsd←path,'\'

vsc←'%comspec% /C ""',vsbat,'" amd64'
vsc,←' && cd "',wsd,'\rtm"'
vsc,←' && cl /MP /W3 /wd4102 /wd4275 /Od /Zc:inline /Zi /FS'
vsc,←'        /Fo".\\" /Fd"codfns.pdb"'
vsc,←'        /WX /MD /EHsc /nologo /I"%AF_PATH%\include"'
vsc,←'        /D "NOMINMAX" /D "AF_DEBUG" /D "EXPORTING"'
vsc,←'        "*.c" /link /DLL /OPT:REF'
vsc,←'        /INCREMENTAL:NO /SUBSYSTEM:WINDOWS'
vsc,←'        /LIBPATH:"%AF_PATH%\lib"'
vsc,←'        /DYNAMICBASE "af',codfns.AF∆LIB,'.lib"'
vsc,←'        /OPT:ICF /ERRORREPORT:PROMPT'
vsc,←'        /TLBID:1 /OUT:"codfns.dll""'
```

This code is used in chunk 39b.
Defines:
`vsbat`, used in chunks 29 and 39c.
`wsd`, used in chunks 39c and 41.
Uses `AF∆LIB` 7, `codfns` 4, and `VS∆PATH` 8.

Finally, in order to write up the test harness to work right, we must copy the appropriate runtime files into the `tests\` directory so that we can find them when we finally start running our code there.

41 ⟨*Copy the runtime files into* `tests\` 41⟩≡

```
]CMD ]←vsc
]CMD ]←'copy "',wsd,'rtm\codfns.h" "',wsd,'tests\"'
]CMD ]←'copy "',wsd,'rtm\codfns.exp" "',wsd,'tests\"'
]CMD ]←'copy "',wsd,'rtm\codfns.lib" "',wsd,'tests\"'
]CMD ]←'copy "',wsd,'rtm\codfns.pdb" "',wsd,'tests\"'
]CMD ]←'copy "',wsd,'rtm\codfns.dll" "',wsd,'tests\"'
```

This code is used in chunk 39b.
Uses `codfns` 4 and `wsd` 40b.

8.3 Loading the Compiler

In order to load the compiler into an APL session as well as all the development utilities, we assume that you have first managed to either load up a session with a bootstrapped version of the `TANGLE` command or that you already have a tangled `src\` directory. If the `src\` directory has not yet been created by running the `TANGLE` command, then this must be done before loading the compiler system. After tangling, the compiler can be loaded using the provided `LOAD` shortcut. This shortcut is meant to use the Dyalog Link system for hot-loading the files in `src\` into the root namespace. We do so through the following link command:

```
Link.Create # src -source=dir -watch=dir
```

This means that we want to link the `src\` directory into the `#` namespace, but we also want to make sure that we only pull changes that come from the filesystem. This is because we are editing the code via the WEB document, and we do not want to risk having some intermediate representation that isn't accurate and that doesn't flow the right way; we want all appropriate changes to begin in the WEB document and then, and only then, flow into the session. This also allows us to make some modifications to the code for testing and experimentation inside of the session without consideration for the code outside of the session, and such changes will be removed or forgotten on the next `TANGLE` command.

To set this up, we also ensure that we begin our work within the root Co-dfns repository directory, as this is where we expect to run the `TANGLE` and `WEAVE` commands.

There is unfortunately only a limited range of possibilities for linking in a new directory as we wish to do. The method we choose to use is launching a fresh Dyalog APL session and then using an `LX` expression from the command line to do the actual linking using the `⎕SE.UCMD` functionality. I personally find this to be rather hackish, and I hope that an alternative approach to doing this will show up in the near future. Nonetheless, the arguments that we pass to `dyalog.exe` look something like this:

```
LX="⎕SE.UCMD'Link.Create # src -source=dir -watch=dir'"
```

If you do not use the `LOAD` shortcut, you can use the above command to do the linking manually.

9 Index

9.1 Chunks

9.2 Identifiers

10 GNU AFFERO GENERAL PUBLIC LICENSE

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU Affero General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user

commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the

stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for per-

sonal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that

is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you per-

mission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in

connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to

time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA

BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>        <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.    See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.      If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a “Source” link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see `https://www.gnu.org/licenses/`.