Empowering DSLs with Automated Language Server Generation

Automatically Generating LSP servers from DSL Specifications

M*N / M+N problem

https://microsoft.github.io/

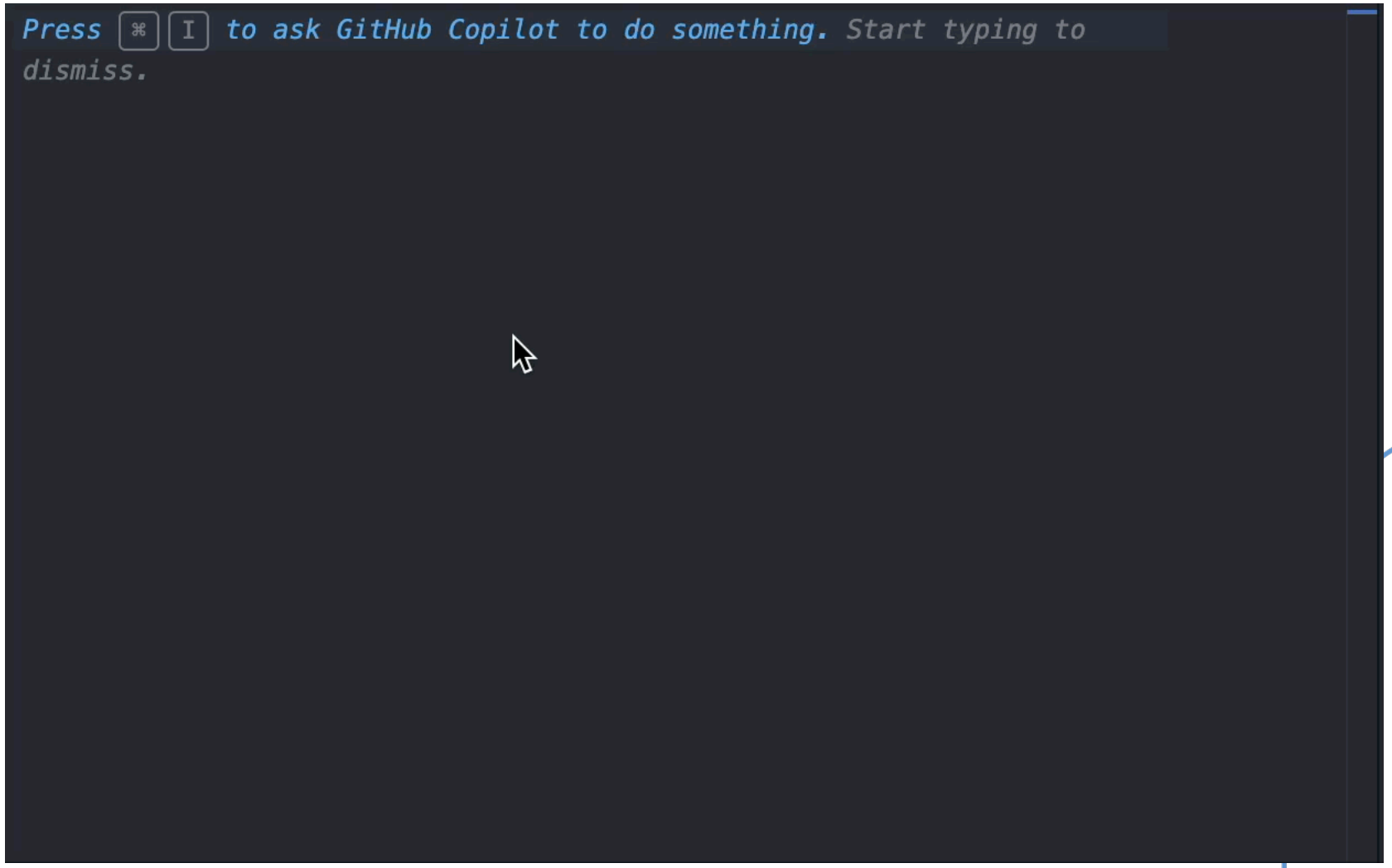
LSP Features

initialize
initialized
shutdown
exit
window/showMessage
window/showMessageRequest
window/logMessage
window/workDoneProgress/create
window/workDoneProgress/cancel
telemetry/event
client/registerCapability
client/unregisterCapability
workspace/didChangeWorkspaceFolders
workspace/didChangeConfiguration
workspace/didChangeWatchedFiles
workspace/symbol
workspace/executeCommand
workspace/applyEdit
workspace/configuration
workspace/workspaceFolders
workspace/codeLens/refresh

workspace/inlayHint/refresh
workspace/semanticTokens/refresh
workspace/diagnostic/refresh
textDocument/didOpen
textDocument/didChange
textDocument/willSave
textDocument/willSaveWaitUntil
textDocument/didSave
textDocument/didClose
textDocument/completion
completionItem/resolve
textDocument/hover
textDocument/signatureHelp
textDocument/declaration
textDocument/definition
textDocument/typeDefinition
textDocument/implementation
textDocument/references
textDocument/documentHighlight
textDocument/documentSymbol
textDocument/codeAction
codeAction/resolve
textDocument/codeLens
codeLens/resolve
textDocument/codeLens/refresh
textDocument/documentLink
documentLink/resolve
textDocument/documentColor
textDocument/colorPresentation
textDocument/formatting
textDocument/rangeFormatting
textDocument/onTypeFormatting
textDocument/rename
textDocument/prepareRename
textDocument/foldingRange
textDocument/selectionRange
textDocument/linkedEditingRange

textDocument/publishDiagnostics
textDocument/semanticTokens/full
textDocument/semanticTokens/full/delt
textDocument/semanticTokens/range
textDocument/prepareCallHierarchy
callHierarchy/incomingCalls
callHierarchy/outgoingCalls
textDocument/inlayHint
inlayHint/resolve
textDocument/prepareTypeHierarchy
typeHierarchy/supertypes
typeHierarchy/subtypes
textDocument/inlineValue
workspace/inlineValue/refresh
textDocument/diagnostic
workspace/diagnostic
workspace/diagnostic/refresh

Typescript - Powerful Language Support



Current State

```
-- Miking is licensed under the MIT license.
-- Copyright (C) David Broman. See file LICENSE.txt
--
-- A simple library that defines map operations over sequences of tuples.

include "option.mc"
include "char.mc"
include "string.mc"

type AssocMap k v = [(k, v)]
type AssocTraits k = {eq : k -> k -> Bool}

-- 'assocEmpty' is an empty associate map
let assocEmpty : all k. all v. AssocMap k v =
  []

-- 'assocLength m' returns the number of key-value pairs in m
let assocLength : all k. all v. AssocMap k v -> Int =
  length

-- 'assocInsert traits k v m' returns a new map, where the key-value pair
-- ('k','v') is stored. If 'k' is already a key in 'm', its old value will be
-- overwritten.
let assocInsert : all k. all v. AssocTraits k -> k -> v -> AssocMap k v -> AssocMap k v =
  lam traits. lam k. lam v. lam m.
    optionMapOrElse (lam. cons (k,v) m)
                    (lam i. set m i (k,v))
                    (index (lam t : (k, v). traits.eq k t.0) m)
```

Metamodel-Driven Grammar Language - Langium

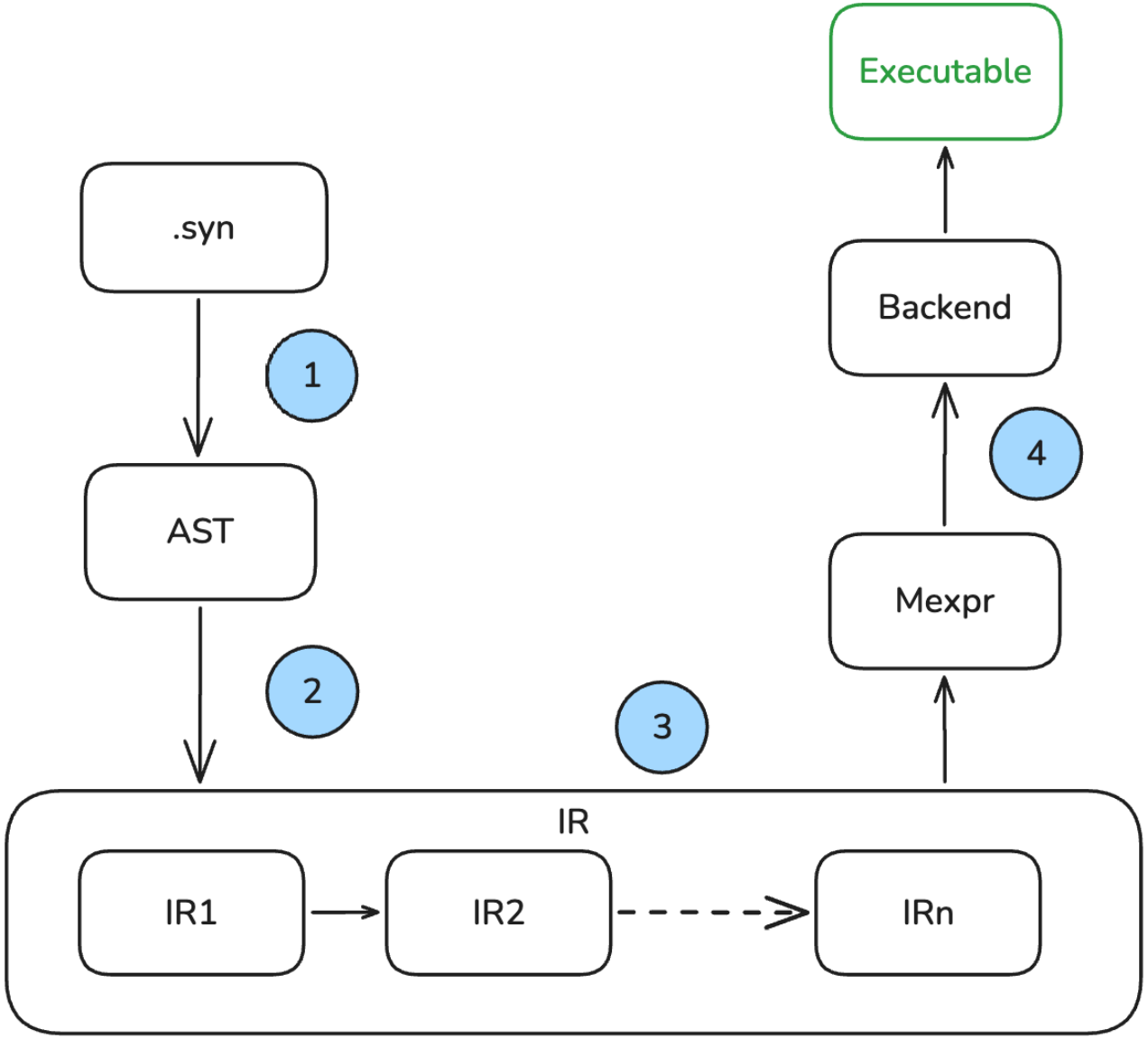
```
Macro:  def=[Def:ID] '(' (args+=Expr (',' args+=Expr)*)? ')';
Def:    'def' name=ID '(' (params+=Param (',' params+=Param)*)? ')' Block;
```

```
Draws the langium logo in MiniLogo!
langium()

/** Draws the langium logo in MiniLogo! */
def langium() {
```

How can we apply this to Mexpr?

- Structural Analysis (1)
- Semantical Analysis (2,3)
- Generated code analysis (4)



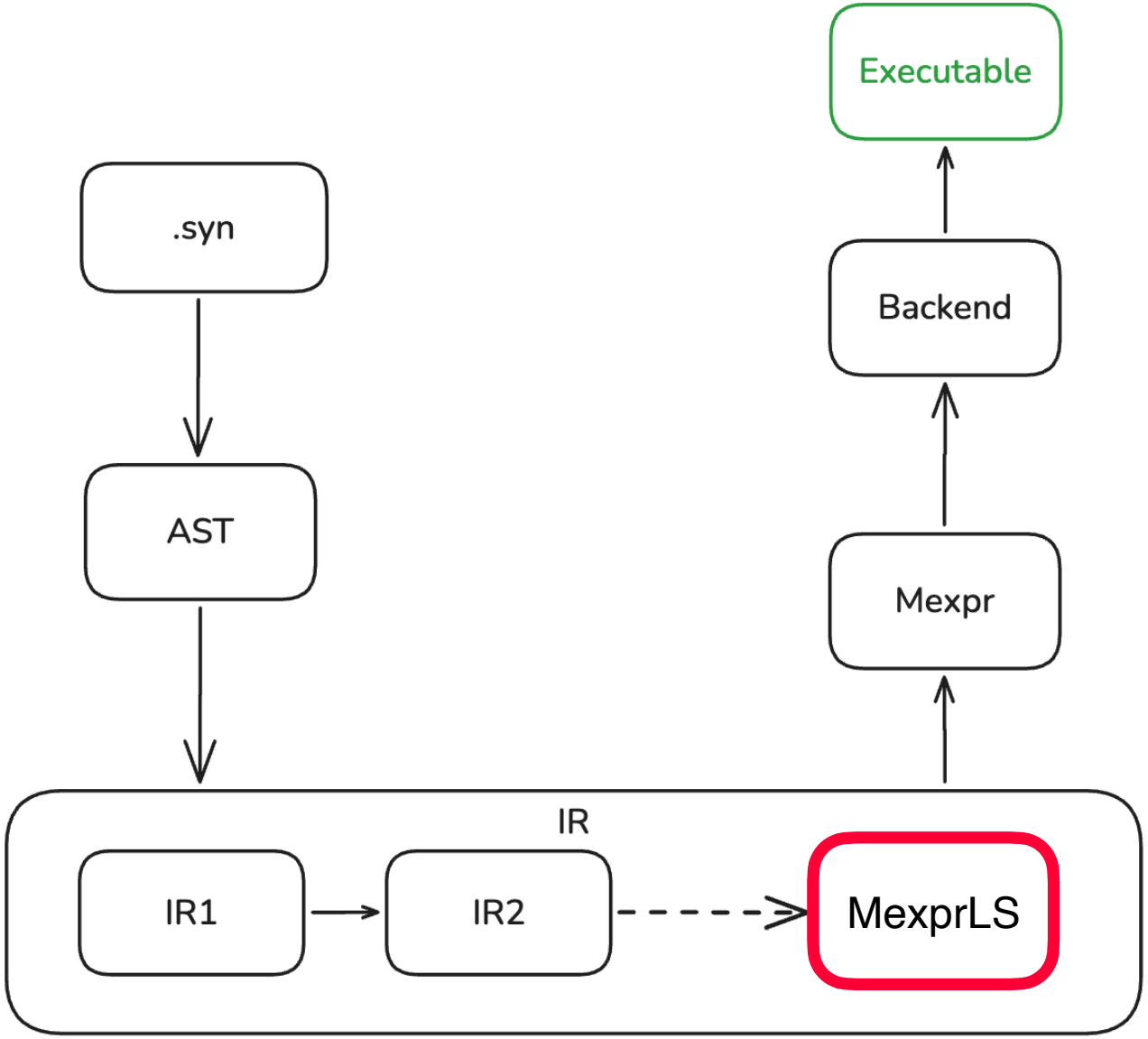
Analysis of Mexpr

```
let myVariable = 100;
print(myVariable);
```

```
TmLet {
  ident = nameNoSym "myVariable",
  body = TmConst {
    val = CFloat {
      val = 100.0
    },
    ty = _tyfloat,
    info = { pos: 57 }
  },
  inexpr = TmApp {
    lhs = TmConst {
      val = CPrint ()
    },
    rhs = TmVar {
      ident = nameNoSym "myVariable",
      info = { pos: 87 },
    },
    info = { pos: 65 }
  }
  ty = _tyfloat x.info,
  info = { pos: 53 }
}
```

Proposal: MexprLS

- Extend common backend with language support constructs



Conclusion

- LSP - Standardized protocol for IDE language support
- Automatic generation of language support by looking at functional backend

Demo time!