# moon-slice 项目申报书

## 基本信息

- 项目名称：moon-slice — MoonBit 程序切片与依赖图工作台
- 参赛者：陈勇润（GitHub：NaIWaniubi）
- 联系方式：见报名表（邮箱 / 手机）
- GitHub 仓库链接：https://github.com/NaIWaniubi/moon-slice
- 发布地址：https://mooncakes.io/docs/NaIWaniubi/moon-slice
- 项目方向：MoonBit 程序分析基础库 / 开发者工具
- 是否为移植项目：否（原创项目，算法参考公开论文）

## 项目简介

moon-slice 是面向 MoonBit 的程序切片与依赖分析工作台。输入一个 `.mbt` 文件（或整个包）与目标语句，它基于 MoonBit 官方 parser 构建控制流图、定义-使用链、控制依赖图与程序依赖图，输出向后切片（这个值从哪来）、向前切片（改这里会影响谁）、死代码与无人调用的私有函数、以及调用点查询，并给出 JSON / DOT / SARIF / 自包含 HTML 报告与浏览器 playground。

项目面向在 MoonBit 中做重构、调试、代码审查与 CI 质量门禁的库作者和团队：把"这次改动影响了什么""哪段代码已经死了"变成可复现的命令行输出和可上传的 SARIF，而不是靠通读代码。

## 核心功能范围

- MoonBit 官方 parser 适配与语句级 IR（`SNode` / `VarId` / `Loc`），覆盖自由函数、`impl` 方法与 `view` 方法体；
- CFG（顺序、分支、循环、跳转）、reaching definitions、控制依赖（结构化与后支配两种）、PDG；
- 向后/向前切片，输出节点集合与源码行号；一行多语句时列出候选并用 `--node` 精确选择；
- 包级过程间切片：`slice --dir D --file F --line N --interprocedural` 跨文件展开被调函数（裸调用展开整个函数体，有返回值的调用只展开返回值 cone）；
- 死代码：`dead`（无使用赋值 + 不可达语句）与 `unused`（整包无人调用的私有函数，尊重 `#warnings("-unused_value")` 标注）；
- 就地抑制：`// moon-slice: ignore[rule]` 写在告警所在行或其上一行，随代码移动，比按行号记录更耐改；
- 反向查询：`callers --dir D --name F` 列出包内全部调用点及所属函数，含限定调用 `T::f(x)`；
- 改动影响分析：`review --diff` 对 git diff 的每个新增行做向前切片，汇总受影响行与函数；
- CI 集成：SARIF 2.1.0（`dead-store` / `unreachable-code` / `unused-function`）、基线抑制 `--baseline`、门禁 `--fail-on-findings`、GitHub composite action；
- 报告与演示：JSON / DOT / 内联 SVG / 自包含 HTML；浏览器 playground 支持点行切片、示例预设与分享链接；
- 质量证据：338 个测试（含独立路径枚举预言器、种子化属性测试、真实仓库回归）；585 文件 / 98,384 行真实代码 0 误报；1987 文件包级存活分析 0 误报；
- 工程化：零警告构建；GitHub Actions CI 覆盖 check / build / test / CLI smoke / playground presets / action 自测；Apache-2.0；`moon package` 打包验证。

## 移植或参考说明

- 原项目名称：无（原创项目，未移植任何现有代码）
- 参考实现：算法参考 Weiser《Program Slicing》(1981)、Ferrante–Ottenstein–Warren《The Program Dependence Graph and Its Use in Optimization》(1987)
- 依赖库：`moonbitlang/parser` 0.4.6、`moonbitlang/x` 0.5.1（均为 Apache-2.0）
- 本项目许可证：Apache-2.0

与已有切片工具相比，本项目做以下取舍：

- 只做静态切片，不做动态切片与符号执行；无法判定的分支按保守方向超近似（宁可多给一行，不漏一行）；
- 过程间展开不做实参到形参的位置映射——MoonBit 有默认参数与具名参数，位置映射会漏掉真实依赖，改为整返回值 cone；
- 方法派发（`x.f()`）仅在接收者类型有标注（参数 / let / impl 的 self）且同包声明该类型方法时解析，且只用于 `callers`；切片展开不跟，避免凭空造边；
- 不重写编译器前端，直接复用官方 parser 的 AST；解析失败按文件报告，不猜测结果。