UFR de mathématique et d'informatique
Université de Strasbourg

MASTER'S DEGREE IN COMPUTER SCIENCE
SCIENCE AND ENGINEERING OF NETWORKS, INTERNET, AND SYSTEMS

Defense presented by

Lucian MOCAN

Internship

Improving the Modularity and Robustness of the Althread Language

August 27, 2025

Supervised by

Anissa LAMANI, Quentin BRAMAS

Agenda

1. Host Institution
2. Problem Context and Importance
3. Contributions
4. Conclusion

1. Host Institution

UFR de mathématique et d'informatique - University of Strasbourg

- Unistra: 6 campuses, 55000 students
- Research & education: 150 members
- ICube
- "Distributed Algorithms" course
- Althread

Image 1. The building of the Department of Mathematics and Computer Science, finally restored in 2024, after 3 years in works [1].

[1] Savoirs. Le bâtiment de l'UFR Math-Info inauguré après 3 ans de travaux. 2024.
URL: https://savoirs.unistra.fr/campus/le-batiment-de-lufr-math-info-inaugure-apres-3-ans-de-travaux

2. Context

Why Althread?

- Model checking and distributed systems verification
- Channels, processes, shared variables, always/never conditions
- Spin (Promela)[2], TLA+[3] (spectacle[4])

```
shared { // variables available anywhere in the code
    let N: int = 0;
    let Start = false;
}
always { // check if conditions are always true
    N == 0;
}
program A() { // program template A
    await Start; // waits until Start == true
    // waits on the process' input channel
    await receive in (x,y) => {
        print("received ");
        // update the value to trigger a violation
        N = N + 1;
    };
}
main {
    // start a process with program template A
    let pa = run A();
    // create and link a channel to A's input
    channel self.out (int, bool)> pa.in;
    Start = true;
    send out (125, true); // send in the channel
}
```

[2] Gerard J. Holzmann. “The model checker SPIN”. In: IEEE Transactions on Software Engineering 23.5 (1997). URL: https://spinroot.com/spin/Doc/ieee97.pdf.
[3] Leslie Lamport. Industrial Use of TLA+. https://lamport.azurewebsites.net/tla/industrial-use.html.
[4] William Schultz. Spectacle: Interactive web-based tool for exploring and sharing TLA+ specifications. GitHub repository; MIT license; 2025. URL: https://github.com/will62794/spectacle.

2. Context

Why Althread?

- Developed at the University of Strasbourg
- Compiler and VM written in Rust, open-source[5]
- Worked on Althread to add support for user-defined functions
- Other contributors: eventually conditions, message flow graph visualization
- Web-based editor

```
fn fibonacci_recursive(n: int, a: int, b: int) -> int {
  if n == 0 {
    return a;
  } else {
    return fibonacci_recursive(n - 1, b, a + b);
  }
}
```

Code snippet 1. A recursive version of Fibonacci that uses a single call

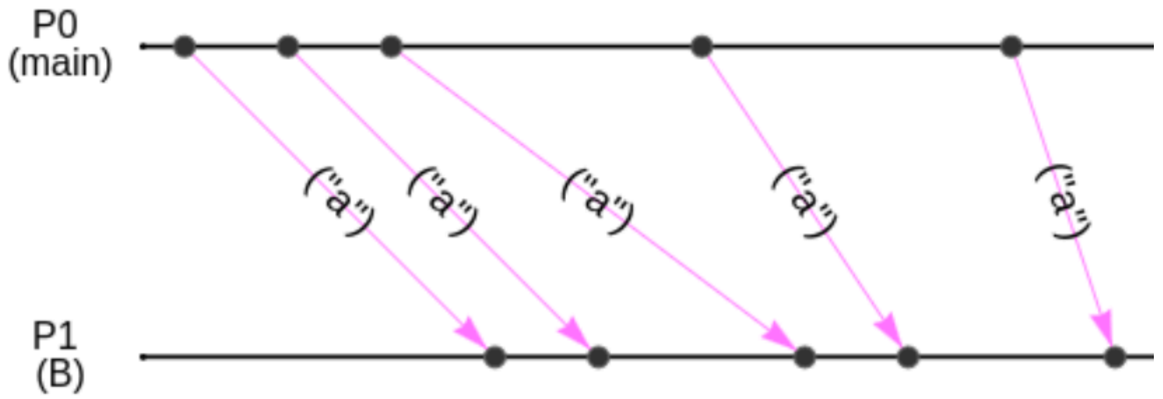


Figure 1. Message flow graph where all messages go from process P0 to P1.

[5] Althread. Introduction to Althread Guide. 2025. URL: https://althread.github.io/en/docs/guide/intro/

2. Context

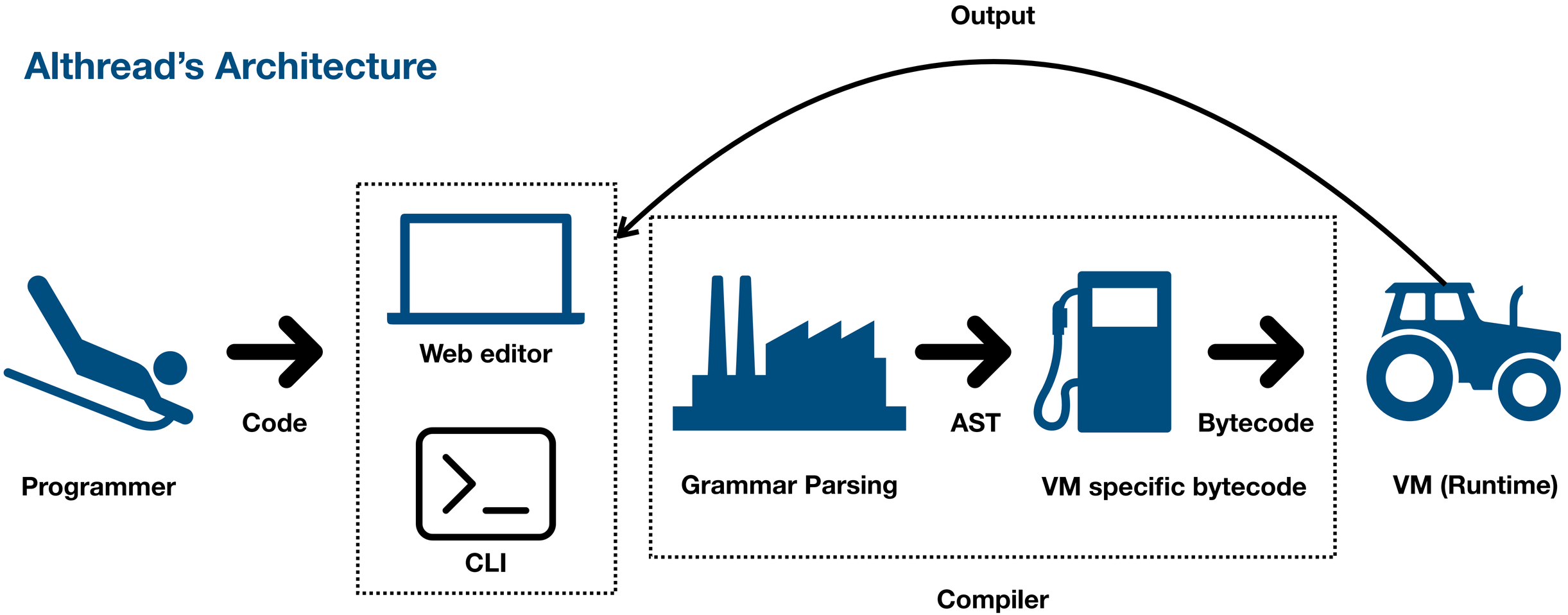
Problem Statement

How to improve Althread so it can be used in teaching distributed algorithms?

- Improve **modularity**:
  - Reuse code, create packages, distribute, contribute
- Improve the **web app**:
  - File explorer, file tabs, package manager, import support
  - Interactive simulated execution
  - Enhance the UI/UX, the tutorials, the documentation

2. Context

Althread's Architecture



AST = Abstract Syntax Tree

VM = Virtual Machine

Bytecode = Instructions that the VM understands

3. Contributions

- Finish / improve work on user-defined functions
- Integrate other TER contributions
- Design and implement an import, module, package system
- Improve the UI/UX experience for the web-based platform

3. Contributions

User-Defined Functions

- Cross-function calling
- Implicit returns

```
fn max(a: int, b: int) -> int {
    if a > b {
        return a;
    } else if a == b {
        // return here ?
    }
    // return here ?
}
```

Code snippet 2. The function **max** has execution paths where no value is returned, violating its declared return type *int*.

3. Contributions

User-Defined Functions

- Cross-function calling
- Implicit returns
- **Control-flow analysis for return path safety**

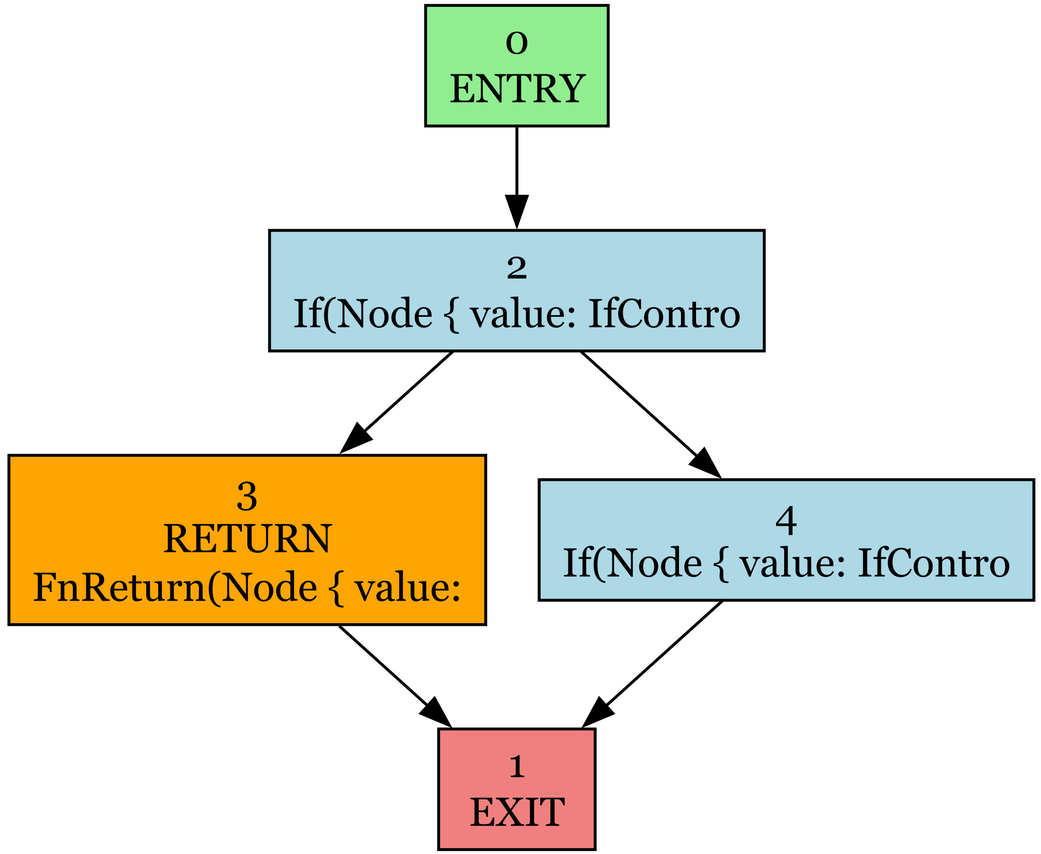


Figure 2. Control-flow graph for the previous **max** function

3. Contributions

Import & Module System

Promela
#include “file.pml” uses gcc preprocessing[6]
No namespace
No visibility modifiers
#ifndef #define #endif guards to prevent circular imports
No package manager

[6] Nimble Code. Promela PreProcessing — main.c (lines 578–614). URL: https://github.com/nimble-code/Spin/blob/master/Src/main.c#L578-L614.

3. Contributions

Import & Module System

Promela	TLA+[7]			
#include “file.pml” uses gcc preprocessing	**EXTENDS** Mod / **INSTANCE** Mod			
No namespace	**EXTENDS** merges namespace; **INSTANCE** creates a separate one (access via *Mod!*)			
No visibility modifiers	**LOCAL** keyword			
#ifndef #define #endif guards to prevent circular imports	No mechanism: duplicate definitions cause errors			
No package manager	None			

[7] Learn TLA+ Guide. Modules (Core section — Learn TLA+). URL: https://learntla.com/core/modules.html.

3. Contributions

Import & Module System

Promela	TLA+	Python[8,9]		
#include “file.pml” uses gcc preprocessing	**EXTENDS** Mod / **INSTANCE** Mod	**import** pkg.mod / **from** pkg **import** name		
No namespace	**EXTENDS** merges namespace; **INSTANCE** creates a separate one (access via *Mod!*)	Modules act as namespaces; accessed as **pkg.mod**		
No visibility modifiers	**LOCAL** keyword	**_private** keyword, etc.		
#ifndef #define #endif guards to prevent circular imports	No mechanism: duplicate definitions cause errors	Detect cycles at import time		
No package manager	None	pip, etc. (external)		

[8] David Beazley. Modules and Packages: Live and Let Die! — PyCon 2015. URL: https://www.youtube.com/watch?v=0oTh1CXRaQ0.
[9] Python Documentation. Standard Modules — Python 3 Tutorial. URL: https://docs.python.org/3/tutorial/modules.html#standard-modules.

3. Contributions

Import & Module System

Promela	TLA+	Python	Go[10]	
#include “file.pml” uses gcc preprocessing	**EXTENDS** Mod / **INSTANCE** Mod	**import** pkg.mod / **from** pkg **import** name	**import (** **"fmt"** **"github.com/user/repo/package"** **)**	
No namespace	**EXTENDS** merges namespace; **INSTANCE** creates a separate one (access via *Mod!*)	Modules act as namespaces; accessed as **pkg.mod**	Package name is a namespace; accessed as **pkg.Func**	
No visibility modifiers	**LOCAL** keyword	**_private** keyword, etc.	Exported identifiers start with a capital letter; others are private	
#ifndef #define #endif guards to prevent circular imports	No mechanism: duplicate definitions cause errors	Detect cycles at import time	Compile-time error for import cycles	
No package manager	None	pip, etc. (external)	**go mod** (built-in)	

[10] The Go Authors. A Tour of Go – Basics. URL: https://go.dev/tour/basics.

3. Contributions

Import & Module System

Promela	TLA+	Python	Go	Althread
#include "file.pml" uses gcc preprocessing	**EXTENDS** Mod / **INSTANCE** Mod	**import** pkg.mod / **from** pkg **import** name	**import (** "**fmt**" "github.com/user/repo/package" **)**	**import** [display **as** d, math, github.com/user/repo/package]
No namespace	**EXTENDS** merges namespace; **INSTANCE** creates a separate one (access via *Mod!*)	Modules act as namespaces; accessed as **pkg.mod**	Package name is a namespace; accessed as **pkg.Func**	Each import becomes a module; accessed via **module.name**
No visibility modifiers	**LOCAL** keyword	**_private** keyword, etc.	Exported identifiers start with a capital letter; others are private	Everything is exported by default; **@private** directive to hide
#ifndef #define #endif guards to prevent circular imports	No mechanism: duplicate definitions cause errors	Detect cycles at import time	Compile-time error for import cycles	Resolver checks for cycles and returns an error
No package	None	pip, etc. (external)	**go mod** (built-in)	built-in package manager (alt.toml)

3. Contributions

Import & Module System

1. Prescan for channel declarations and type inference

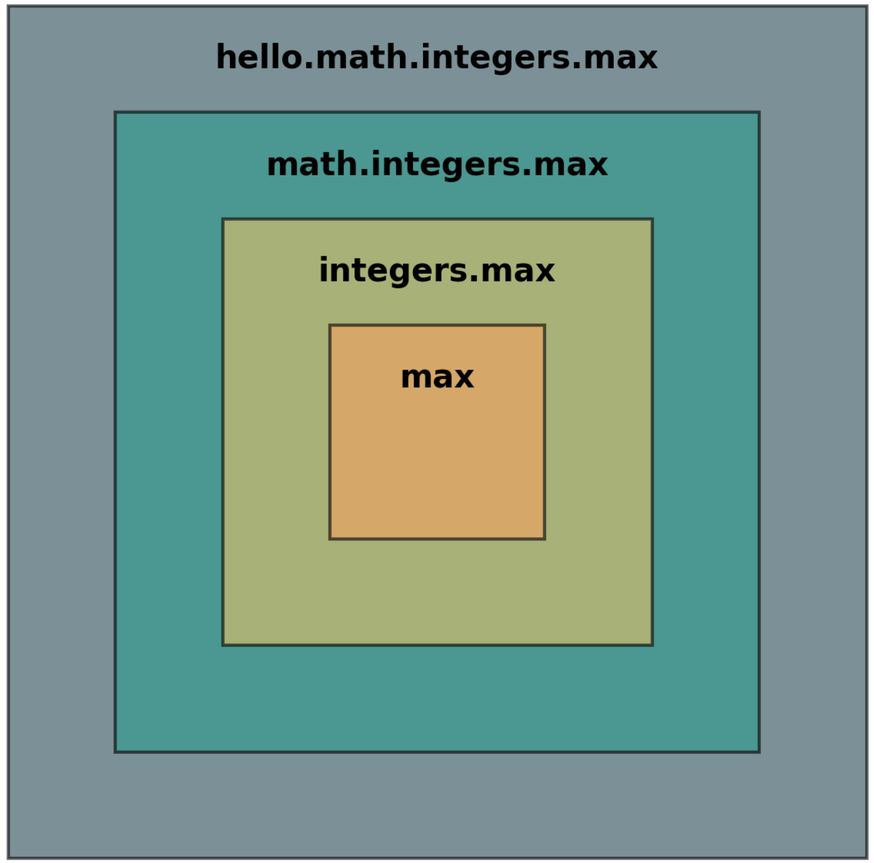


Figure 3. Import name qualifier onion

3. Contributions

Import & Module System

1. Prescan for channel declarations and type inference

2. Compile then qualify all (including imported) functions, programs, shared variable names

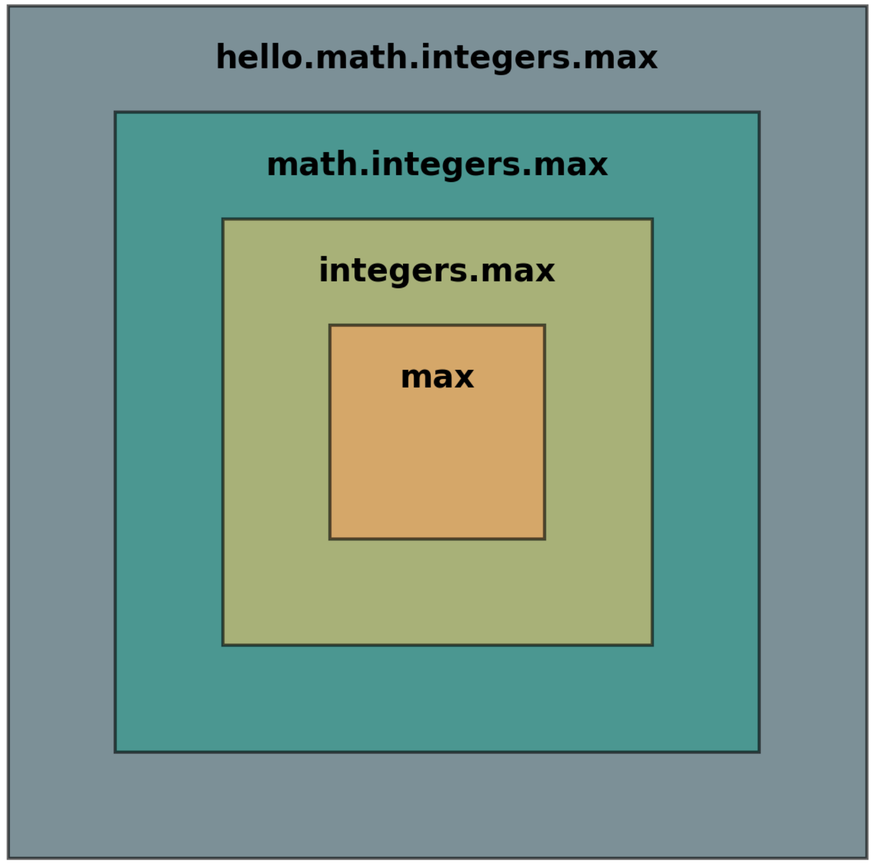


Figure 3. Import name qualifier onion

3. Contributions

Import & Module System

1. Prescan for channel declarations and type inference

2. Compile then qualify all (including imported) functions, programs, shared variable names

3. Post-phase to check privacy constraints

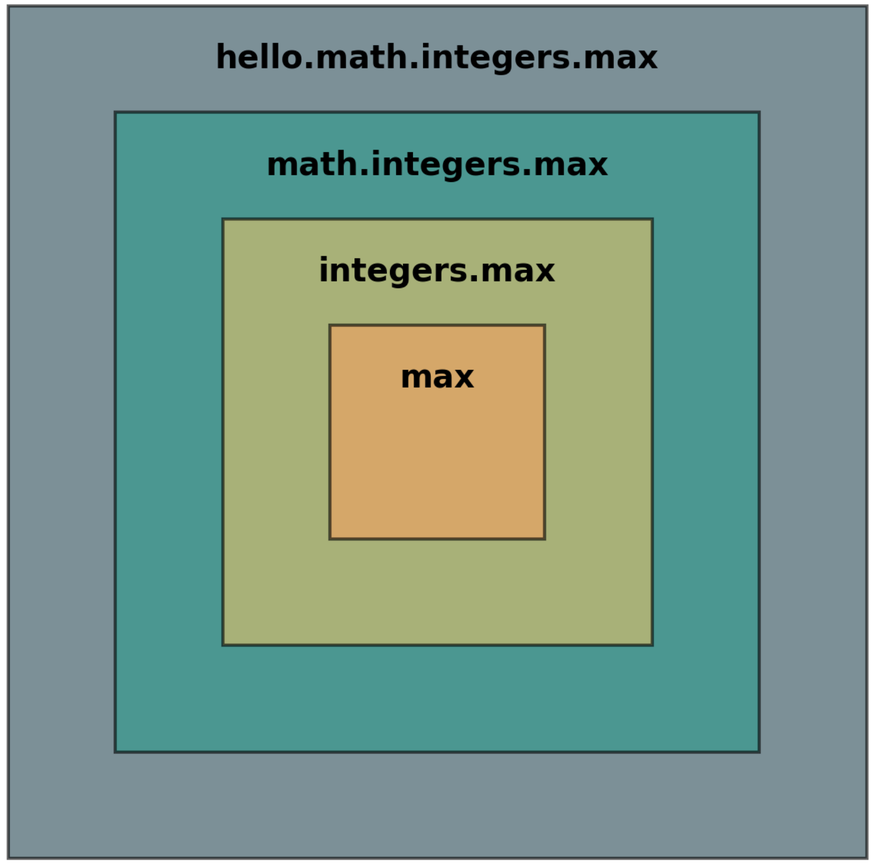


Figure 3. Import name qualifier onion

3. Contributions

Import & Module System

1. Prescan for channel declarations and type inference
2. Compile then qualify all (including imported) functions, programs, shared variable names
3. Post-phase to check privacy constraints
4. Return to the importer

```
import [
    math,
    cool/fib,
    display as d
]
main {
    print(math.max(7,3));
    print(fib.N);
    print(fib.fibonacci_iterative_N());
    fib.N = 10;
    print(fib.fibonacci_iterative_N());
    // print(fib.fibonacci_iterative(fib.N, 0, 1));
    run d.Hello();
}
```

Code snippet 3. Showcase of import use in Althread

3. Contributions

Virtual Filesystem

- Store files in the web: virtual filesystem
- Extend the module resolver
- Error messages with correct path structure

```
althread-file-system: [
    {
        "id":"e0b971c7-bba1-42f2-9830-b6a876ccc101",
        "name":"utils",
        "type":"directory",
        "children":[]
    },
    {
        "id":"ab4c6b89-26f4-45ec-97d8-c2c9e33387e7",
        "name":"main.alt",
        "type":"file"
    }
]
althread-file-content-main.alt: "import [utils/math] ..."
```

Figure 4. What the stored virtual filesystem looks like in localStorage

3. Conclusion

Current and perspectives

- User-defined functions
  - Maybe add support for chained calls
- Import & package system
  - Web & CLI package manager
  - Test the package system
- Documentation (functions, import & package system)
- Step-by-step interactive simulated execution
- Add LTL block for model checking richness

3. Conclusion

Current and perspectives

- User-defined functions
  - Maybe add support for chained calls
- Import & package system
  - Web & CLI package manager
  - Test the package system
- Documentation (functions, import & package system)
- Step-by-step interactive simulated execution
- Add LTL block for model checking richness

234 commits, **196** files modified: **35,082** additions and **6,206** deletions.

3. Conclusion

What I learned

- Remote internships are not easier (though there’s more flexibility).
- It takes time to design & research, it takes longer to build.
- Most of the time spent debugging comes down to simple errors.

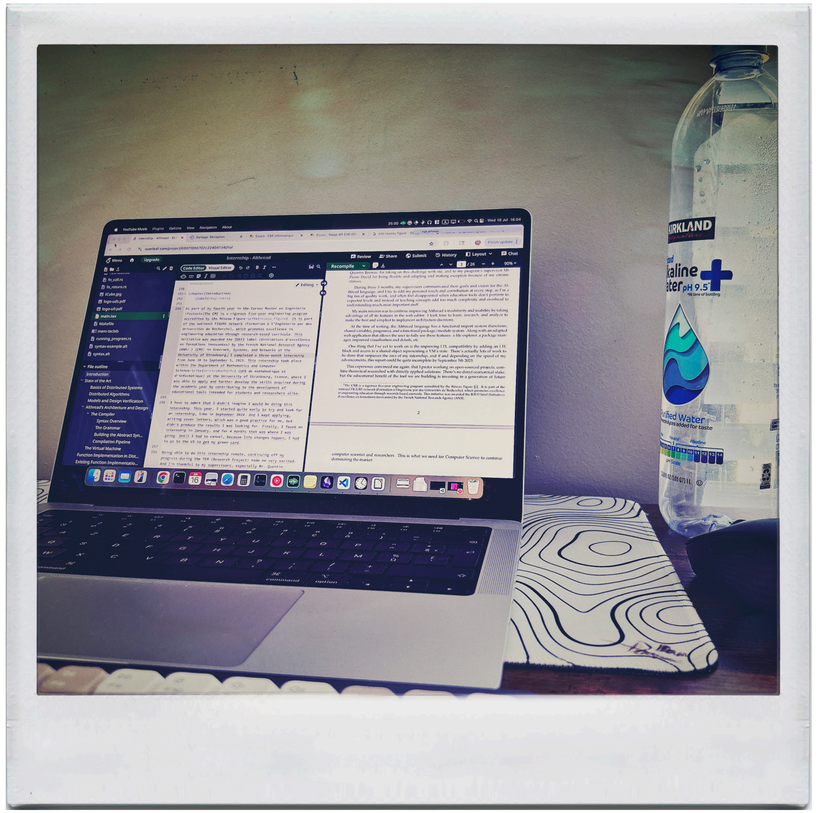


Image 2. The internship workspace

References

[1] Savoirs. Le bâtiment de l'UFR Math-Info inauguré après 3 ans de travaux. 2024.
URL: https://savoirs.unistra.fr/campus/le-batiment-de-lufr-math-info-inaugure-apres-3-ans-de-travaux

[2] Gerard J. Holzmann. “The model checker SPIN”. In: IEEE Transactions on Software Engineering 23.5 (1997).
URL: https://spinroot.com/spin/Doc/ieee97.pdf.

[3] Leslie Lamport. Industrial Use of TLA+. https://lamport.azurewebsites.net/tla/industrial-use.html.

[4] William Schultz. Spectacle: Interactive web-based tool for exploring and sharing TLA+ specifications. GitHub repository; MIT license; 2025. URL: https://github.com/will62794/spectacle.

[5] Althread. Introduction to Althread Guide. 2025. URL: https://althread.github.io/en/docs/guide/intro/

[6] Nimble Code. Promela PreProcessing — main.c (lines 578–614). URL: https://github.com/nimble-code/Spin/blob/master/Src/main.c#L578-L614.

[7] Learn TLA+ Guide. Modules (Core section — Learn TLA+). URL: https://learntla.com/core/modules.html.

[8] David Beazley. Modules and Packages: Live and Let Die! — PyCon 2015. URL: https://www.youtube.com/watch?v=0oTh1CXRaQ0.

[9] Python Documentation. Standard Modules — Python 3 Tutorial. URL: https://docs.python.org/3/tutorial/modules.html#standard-modules.

[10] The Go Authors. A Tour of Go – Basics. URL: https://go.dev/tour/basics.