CerCo Work Package 5
Task 5.1: Management of complexity assertions
Task 5.3: Case studies

Nicolas Ayache

Post-doc at University Paris Diderot — PPS — INRIA πr2

March 2012, the 16th

Objectives

Task 5.1: Management of complexity assertions

$$\text{C} \xrightarrow{\text{CerCo}} \text{C annotated} \xrightarrow{?} \text{WCET of each function (VCs)}$$

Task 5.3: Case studies

$$\text{Lustre} \xrightarrow{?} \text{certified component reaction time}$$

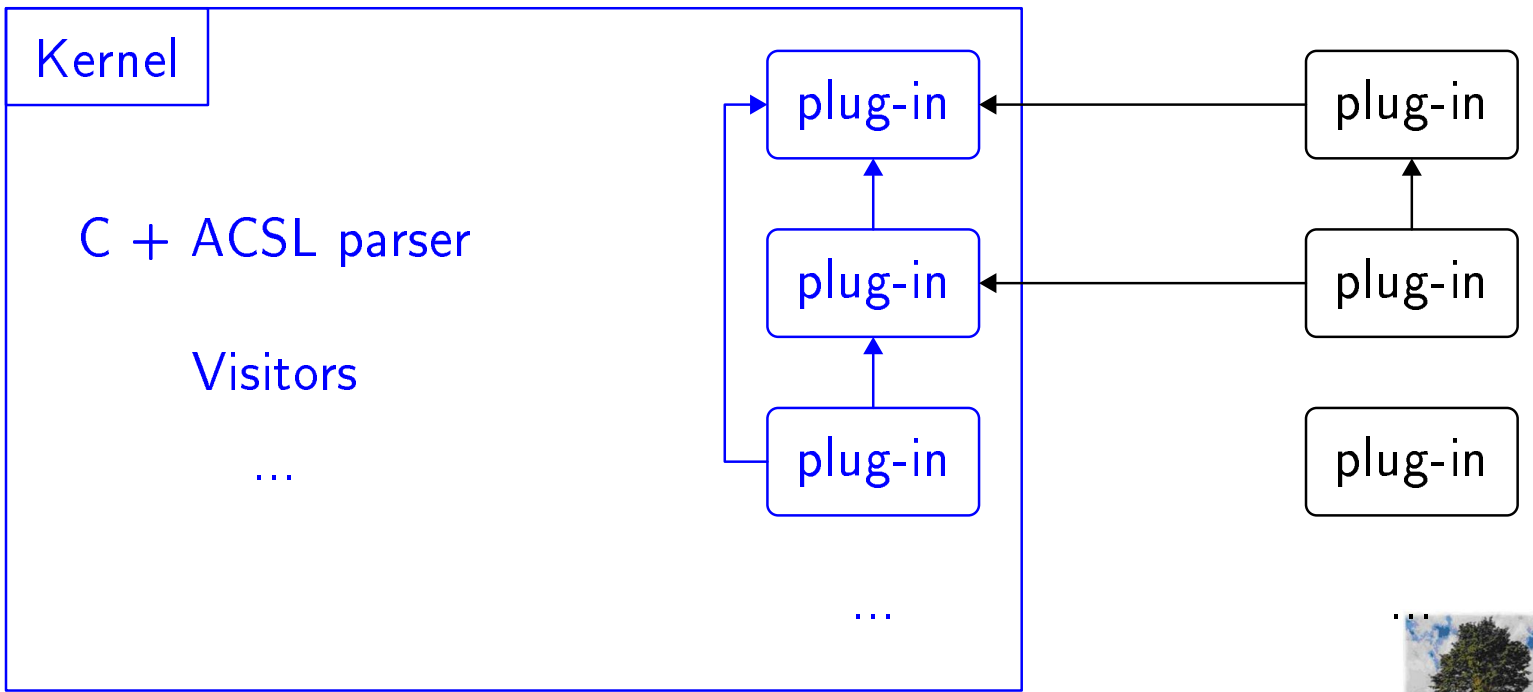
SEVENTH FRAMEWORK PROGRAMME

Outline

Outline

Architecture

Tool for collaborative and modular analysers for C



SEVENTH FRAMEWORK PROGRAMME

Jessie plug-in

ACSL

Specification language for C programs

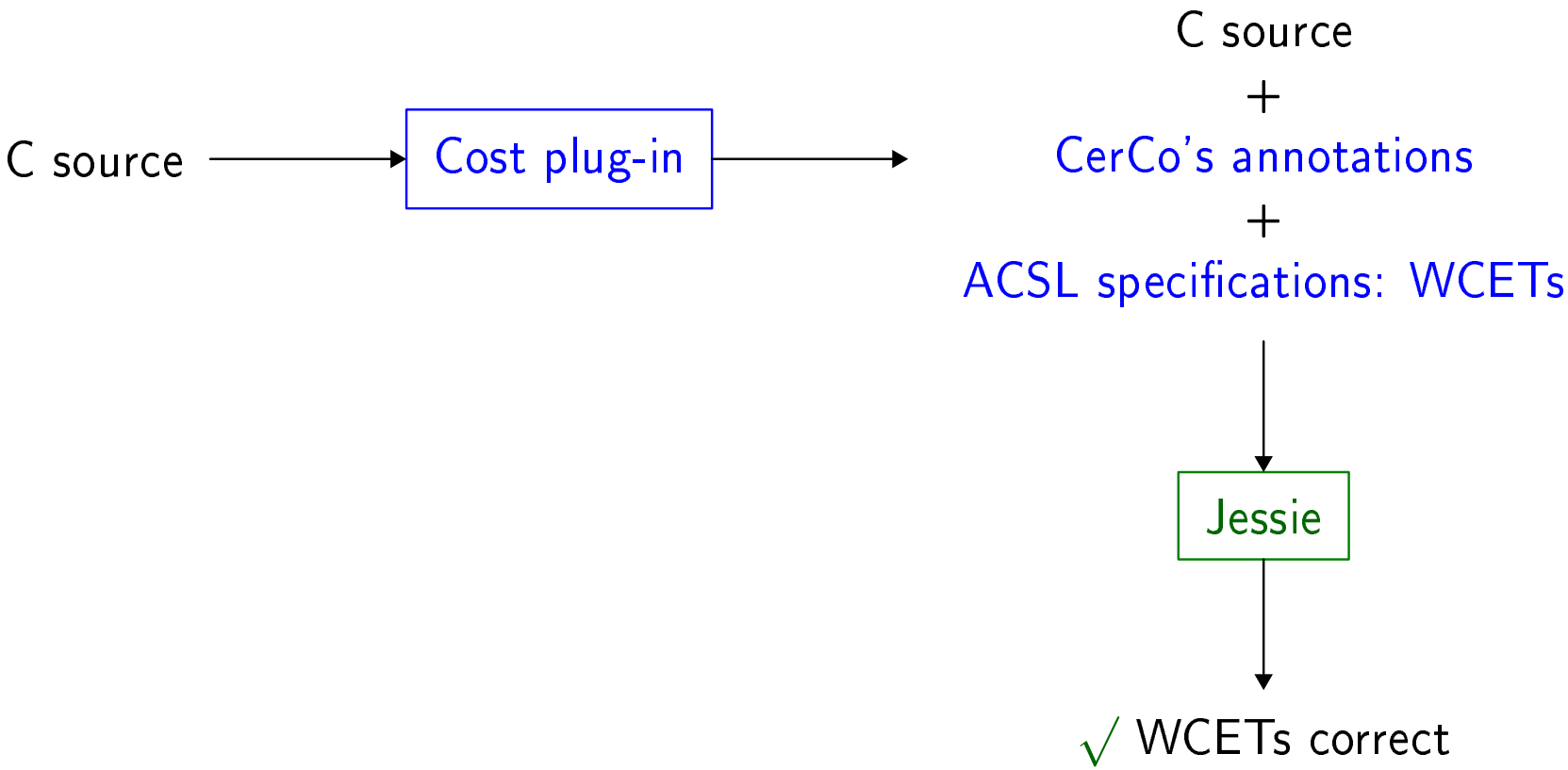
- Based on Hoare logic
- C Comments
- First-order logical definitions and propositions

Jessie

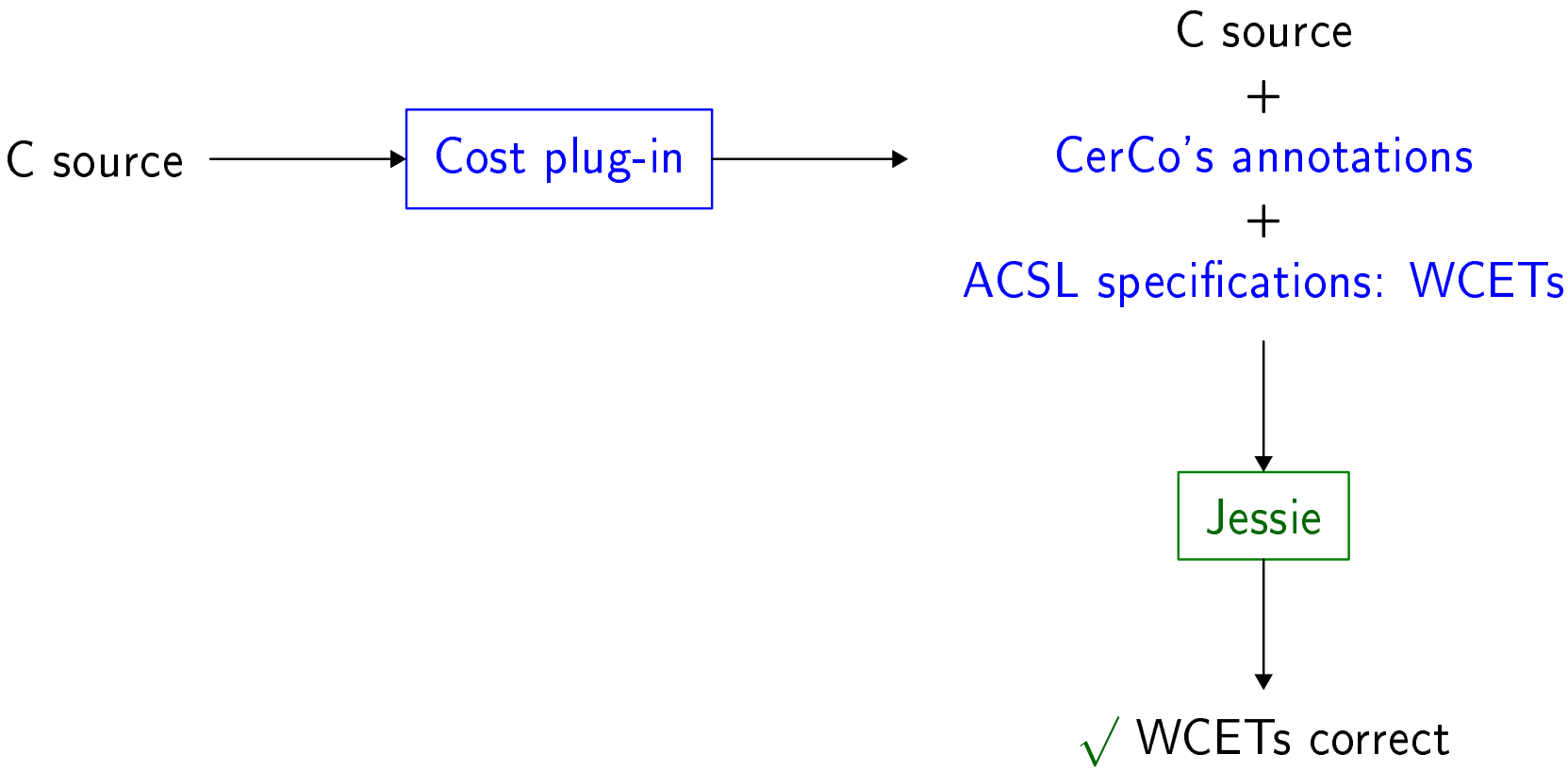
Translator to Why

- Weakest pre-condition calculus
- Generates proof obligations
- Output to various provers (automatic or interactive)

Usage

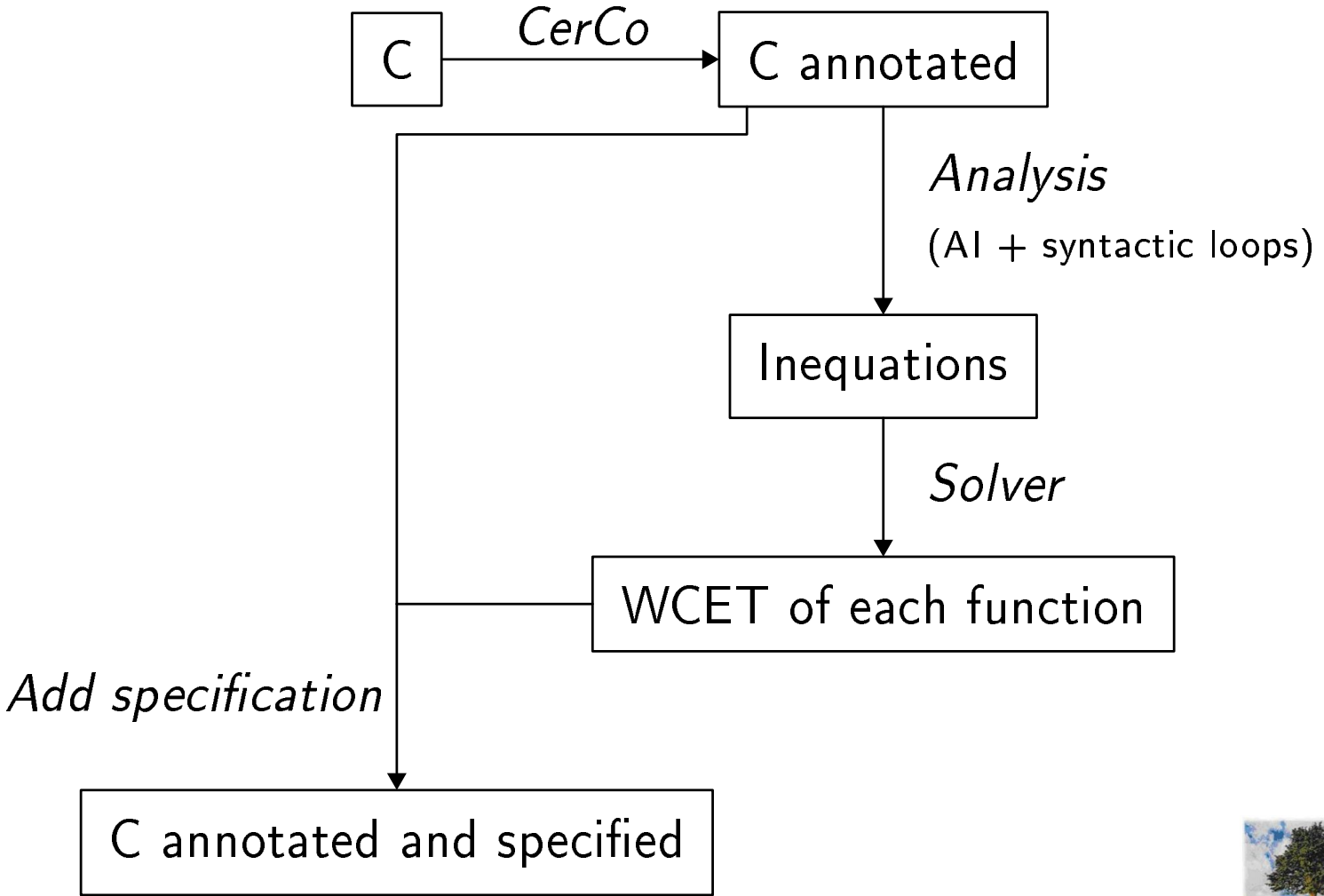

C source
Cost plug-in
C source
+
CerCo's annotations
+
ACSL specifications: WCETs
Jessie
√ WCETs correct

Usage


C source
Cost plug-in
C source
+
CerCo's annotations
+
ACSL specifications: WCETs
Jessie
√ WCETs correct

Demo

Architecture



Analysis

Function call

Symbolic

Example: `f(tab, size)` $\longrightarrow$ *cost of* `f(`$[\![$`tab`$]\!]^\sharp$`, `$[\![$`size`$]\!]^\sharp$`)`

Loop

Cost of the loop relative to the cost before the loop

Example:

```
_tmp_loop0 = _cost;
/*@ loop invariant _cost ≤ _tmp_loop0 + i * body cost;
  @ ...  *@/
for (i = 0 ; i < size ; i++) { body }
```

Cost: replace current iteration with total iteration

Inequations

Form

Example:

$$\begin{aligned} \textit{cost of } \texttt{f(tab, size)} &\leq 5 + \texttt{size} \times \textit{cost of } \texttt{g(size)} \\ \textit{cost of } \texttt{g(size)} &\leq 10 \times \texttt{size} \end{aligned}$$

Arithmetic expressions with:

- Formal parameters
- Cost of other functions

Solution

Fixpoint:

- Replace the cost of solved (independent) functions
- X May not find a solution (e.g. recursive functions)

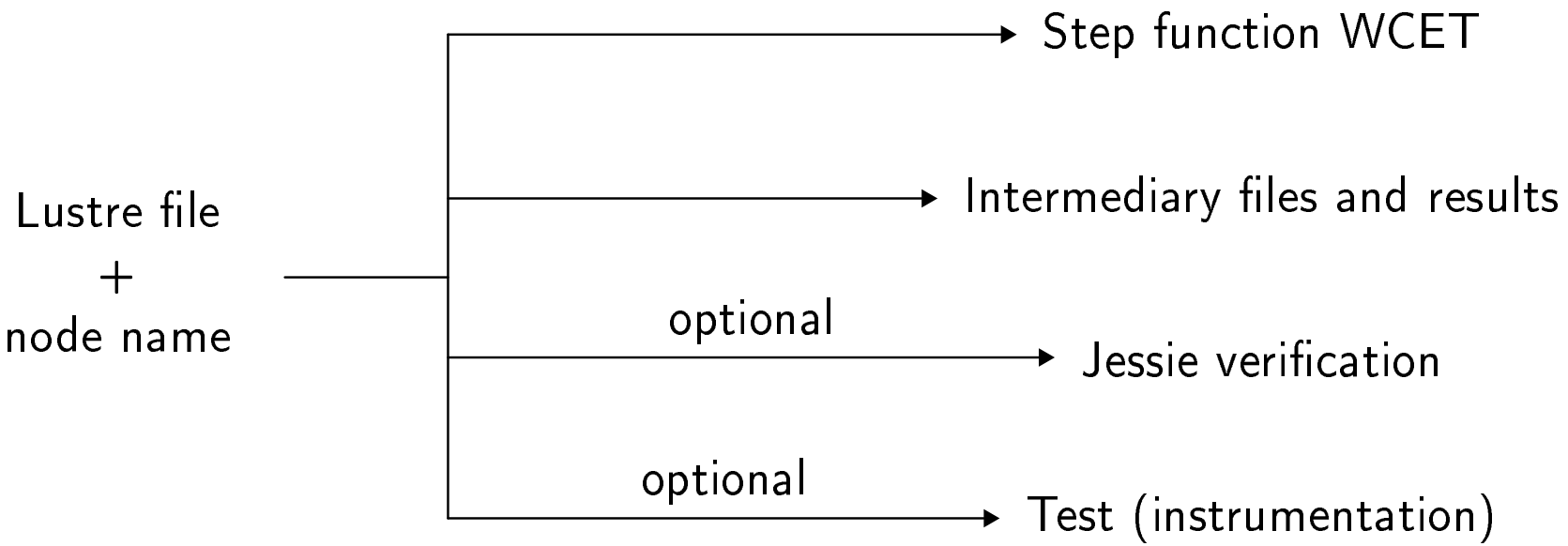
Outline

- Synchronous language
- Node = flow
- `lus2c`: node $\rightarrow$ C step function (cycle)
- Step function WCET = component reaction time

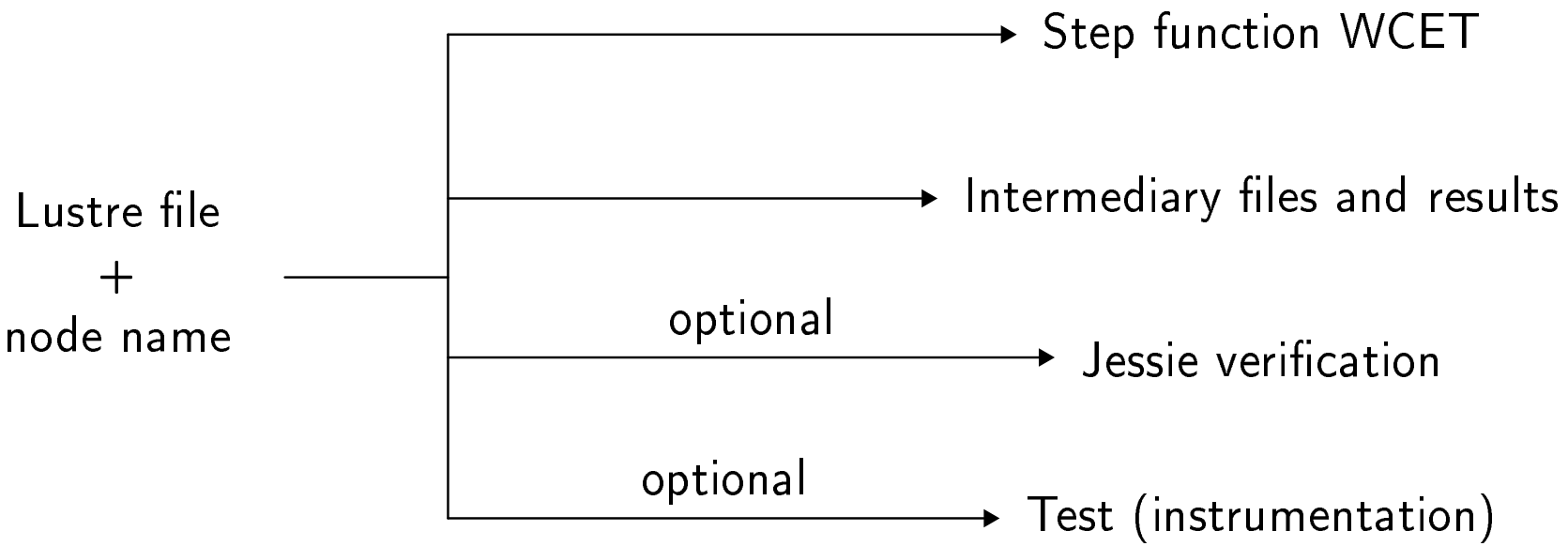
Lustre C specificities

- No arrays
- No loop
- Boolean variables identified

Usage

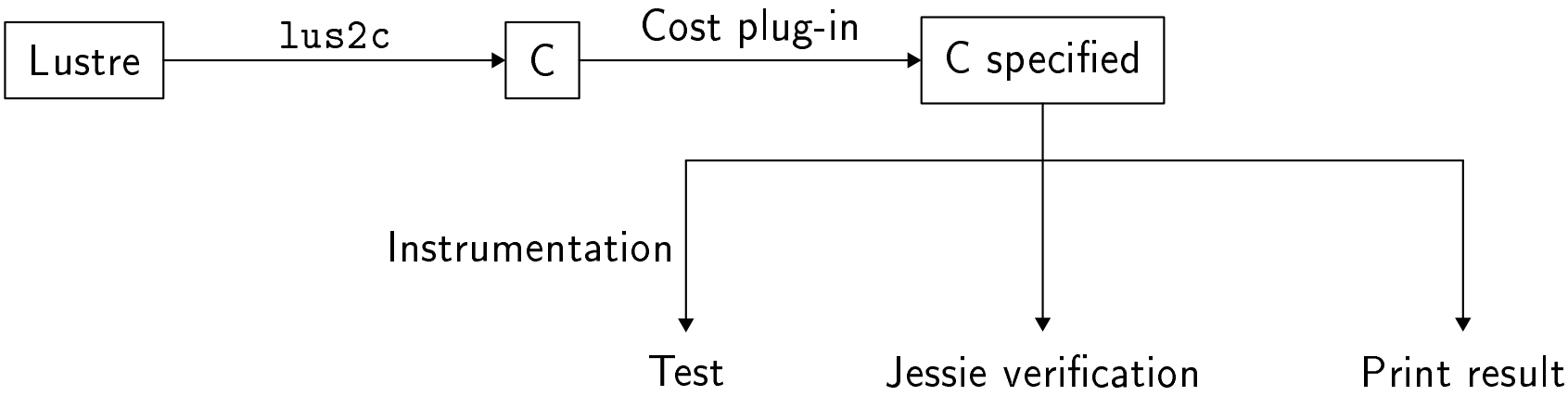

Lustre file
+
node name
Step function WCET
Intermediary files and results
optional
Jessie verification
optional
Test (instrumentation)

Usage


Lustre file
+
node name
Step function WCET
Intermediary files and results
optional
Jessie verification
optional
Test (instrumentation)

Demo

Architecture



Test

- Interpretation in Clight
- Random initial states, ran for some cycles

Perspectives

- More loops recognized
- Better abstract value domain
- More code examples
- More hints for provers