

Getting Started with Clemson's Verifying Compiler

From the CSSE373 Moodle Site

Week #9 - 11 May



Topics:

1) Navigate to Week #9
373 Moodle Site


- Announcement:
 - Given the unusual circumstances in the world today and the extra stress this has created ...
 - Exam 3 will be replaced with online activities (TBA below) that will have you use an experimental Verifying Compiler
 - ProjP3 will not be assigned
 - **ProjP1** and **ProjP2** deadlines have been extended from 15 May to 19 May (additional 4 days)
- Professor Tony Hoare's Verifying Compiler Grand Challenge
- Using the RESOLVE Verifying Compiler

2) Click on Resolve Verifying Compiler

Slides:

- **Link to RSRG Website and the RESOLVE Verifying Compiler**
- **Hoare's Paper Outlining the Grand Challenge**
- Video and Slides TBA

Takes You to RESOLVE Research Group's Landing Page

The screenshot shows a web browser window with the address bar displaying `cs.clemson.edu/resolve/`. The page features the **RSRG** logo and the text **RESOLVE Software Research Group**. A navigation bar contains links for Home, About, Research, Teaching, and Web IDE. A red box with the text "1) Click on Web IDE tab" and a red arrow points to the Web IDE link. To the right of the navigation bar is the NSF logo. The main content area begins with a welcome message: "Welcome to the Clemson RESOLVE Software Research Group (RSRG) website." This is followed by a paragraph describing the RESOLVE research effort, its vision, and its funding sources (NSF, U.S. Department of Education, and NASA). Below this, another paragraph states: "The RESOLVE verification vision is indeed ambitious. We welcome professors, practitioners, and students to contribute and get involved in realizing the vision. The RESOLVE project is a  project. Click [HERE](#) to contact us." The footer contains the Clemson University logo, a GitHub logo with a note about HTML5 support, and the Clemson School of Computing logo.


RSRG
RESOLVE Software Research Group

1) Click on Web IDE tab


Home About Research Teaching Web IDE

Welcome to the Clemson RESOLVE Software Research Group (RSRG) website.

The RESOLVE research effort is one of the longest running software engineering efforts in the USA. It spans foundational, practical, and educational aspects of software engineering and computing. The overall RESOLVE [verification vision](#) is that of a future in which no production software is considered properly engineered unless it has been fully specified, and fully verified as satisfying these specifications. The RSRG is directed by [Murali Sitaraman](#), from the [School of Computing](#) at [Clemson University](#). The RESOLVE effort is funded in part by grants from the U. S. National Science Foundation (NSF), the U. S. Department of Education, and the U. S. National Aeronautics and Space Administration (NASA).

The RESOLVE verification vision is indeed ambitious. We welcome professors, practitioners, and students to contribute and get involved in realizing the vision. The RESOLVE project is a  project. Click [HERE](#) to contact us.

CLEMSON
UNIVERSITY

 This site is optimized for modern browsers supporting the HTML5 standard. Click [HERE](#) for the deprecated version. (no longer maintained as of 12/22/11)

CLEMSON
School of COMPUTING

Now You Are on the Web IDE Page

The screenshot shows a web browser window with the title 'RSRG: WebIDE' and the URL 'cs.clemson.edu/resolve/webide.html'. The page features the RSRG logo and a navigation menu with links to Home, About, Research, Teaching, and Web IDE. The main content area includes a section for SIGCSE 2019 with a link to the ACM SIGCSE 2019 page, and a section for Web IDE Versions with three links: 'Begin to Reason', 'Begin to Reason (Minimal Syntax Version)', and 'Reason with Components'. A red box highlights the 'Reason with Components' link, and a red arrow points to it from a text box that says '1) Click on the Reason with Components link'.

RSRG
RESOLVE Software Research Group

Home About Research Teaching Web IDE

SIGCSE 2019

- [ACM SIGCSE 2019](#)

Web IDE Versions

- [Begin to Reason](#)
- [Begin to Reason \(Minimal Syntax Version\)](#)
- [Reason with Components](#)

1) Click on the Reason with Components link

Now You Are in the RESOLVE Verifying Compiler

The screenshot shows a web browser window with the URL `resolve.cs.clemson.edu/teaching`. The browser tab is labeled "Default_Project". The page content includes a welcome message "Welcome User. Current project: Default_Project" and two buttons: "Projects" and "Components". A red arrow points from the "Components" button to a red box containing the text "1) You are now in the IDE for the RESOLVE Verifying Compiler". Another red arrow points from a red box containing the text "2) Click on the Components button" to the "Components" button. On the left side of the page, the word "TEACHING" is written vertically in large red letters. Below the buttons, there is a code editor with the following text:

```
1 -- Please click on Components button.
2 -- Select a program or a (reusable) concept.
3 -- For additional information, please click on Help at top right.
4 -- Please Register and Sign In (top right) in order to create new components.
5   To create a new component, use the right mouse click in the finder.
```

Now Bring Up the Component Finder Dialog

The screenshot shows a web browser window with the URL `resolve.cs.clemson.edu/teaching`. The page title is "Default_Project". The main content area displays a welcome message: "Welcome User. Current project: Default_Project". Below this message are two buttons: "Projects" and "Components". A red arrow points from a text box to the "Components" button. The text box contains the instruction: "1) After clicking the Components button the Component Finder dialog will appear".

On the left side of the browser window, the word "TEACHING" is written vertically in large red letters.

Below the welcome message, there is a code editor with the following text:

```
1 -- Please click on Components button.
2 -- Select a program or a (reusable) concept.
3 -- For additional information, please click on Help at top right.
4 -- Please Register and Sign In (top right) in order to create new components.
5   To create a new component, use the right mouse click in the finder.
```

The "Component Finder" dialog is open in the foreground. It has a title bar with a close button. Inside the dialog, there are two sections: "Programs" and "Concepts", each with a right-pointing arrow next to it. A red arrow points from a text box to the "Programs" section. The text box contains the instruction: "2) Click on Programs".

Now Select Activities to Work On

The screenshot shows a 'Component Finder' dialog box. On the left, there is a sidebar with two items: 'Programs' (highlighted in orange) and 'Concepts' (highlighted in green with a right-pointing arrow). To the right of the sidebar is a list of activities. Red arrows and text boxes provide instructions: 1) An arrow points from the 'Programs' sidebar item to a text box stating '1) After clicking on Programs, a list of Activities will appear'. 2) An arrow points from the first item in the list, 'Activity_101_Reasoning_about_Assignments', to a text box stating '2) Click on Activity_101'. 3) An arrow points from the close button (an 'x' in a grey box) in the top right corner of the dialog to a text box stating '3) Then click the x to dismiss the dialog'. 4) A text box stating '4) Start to work on the selected activity' is located below the list.

Component Finder [x]

1) After clicking on *Programs*, a list of Activities will appear

Programs	Activity_101_Reasoning_about_Assignments
Concepts >	Activity_102_Reasoning_about_Design_by_Contract
	Activity_103_Reasoning_about_Swapping
	Activity_104_Reasoning_about_If_Statements
	Activity_104a_Reasoning_about_DBC_and_If_Statements
	Activity_104b_Reasoning_about_Max_and_If_Statements
	Activity_104c_Reasoning_about_Design_by_Contract
	Activity_105_Reasoning_about_Loops_and_Recursion
	Activity_201_Reasoning_about_Objects
	Activity_301_Reasoning_with_Stack_Objects
	Activity_302_Reasoning_with_Stack_Enhancements
	Activity_302b_Reasoning_with_Stack_Enhancements

2) Click on Activity_101

3) Then click the x to dismiss the dialog

4) Start to work on the selected activity

Here is Activity_101

TEACHING

Default_Project x +

resolve.cs.clemson.edu/teaching

Welcome User. Current project: Default_Project [Register](#) [Sign in](#) [Help](#)

Projects Components

Activity_101_Reas [X] 1) Here is Activity_101

Build MP-Prove Executable + -

```
1 Facility Activity_101_Reasoning_about_Assignments;
2 uses Integer_Template;
3 -- This is a short intro to logical reasoning through activities
4
5 -- Basics
6 -- Activity 1a: Click on MP-Prove to reason about this code
7 -- Activity 1b: Click on build, generate jar, and execute this code
8
9 -- Guided
10 -- Activity 2a: Uncomment Remember and Confirm assertions by removing -- and MP-Prove (prove)
11 -- Activity 2b: Write a new Confirm assertion about the value of K that is true -- and prove
12 -- Activity 2c: Write a Confirm assertion about the value of K that is false -- and attempt to prove
13
14 -- Inquiry
15 -- Activity 3+: Explore assignment statements (e.g., add or remove) and confirm assertions in various ways
16
17 Operation Main();
18 Procedure
19   Var I, J, K: Integer;
20
21   Read(I);
22   Read(J);
23   -- Remember; -- remembers the values of I and J here as #I and #J
24
25   K := I; -- := is used for assignment
26   I := J;
27   J := K;
28
29   -- Confirm I = #J; --- this is an assertion
30   -- Confirm J = #I;
31
32   Write_Line(I);
33   Write_Line(J);
34 end Main;
35 end Activity_101_Reasoning_about_Assignments;
```

2) Read and follow the instructions
Here, 1a, 1b, 2a, 2b, 2c

3) The RESOLVE language and compiler
allow the programmer to insert a
Remember; statement that tells
the Verifier on what line the #
variables get their values

Here on line 23 is where
#I, #J, and #K get their
incoming values

4) The *Confirm* statement tells the
Verifier to prove the assertion
that follows the *Confirm*

Here is the Verifier at Work

The screenshot shows the 'Activity_101_Reas' window with the 'MP-Prove' button highlighted. The code on the left includes comments about logical reasoning and activities, and a procedure 'Main()' with variables I, J, and K. Three verification conditions (VCs) are generated: VC 0_1 (29), VC 0_2 (30), and VC 0_3 (17). The right panel shows the 'PROVED' status for all three VCs, each with a green checkmark. Red annotations explain the process: clicking 'MP-Prove' generates the VCs; the verifier proves all three, as indicated by the green checkmarks. A note also mentions that the goal and givens are like RT Assumes and that the verifier does not work in exactly the same way as an RT.

Activity_101_Reas

Build MP-Prove Executable

```
1 Facility Activity_101_Reasoning_about_Assignments;
2   uses Integer_Template;
3   -- This is a short intro to logical reasoning through activities
4
5   -- Basics
6   -- Activity 1a: Click on MP-Prove to reason about this code
7   -- Activity 1b: Click on build, generate jar, and execute this code
8
9   -- Guided
10  -- Activity 2a: Uncomment Remember and Confirm assertions by removing
11  -- Activity 2b: Write a new Confirm assertion about the value of K that
12  -- Activity 2c: Write a Confirm assertion about the value of K that i
13
14  -- Inquiry
15  -- Activity 3+: Explore assignment statements (e.g., add or remove) a
16
17  Operation Main();
18  Procedure
19    Var I, J, K: Integer;
20
21    Read(I);
22    Read(J);
23    Remember; -- remembers the values of I and J here as #I and #J
24
25    K := I; -- := is used for assignment
26    I := J;
27    J := K;
28
29    Confirm I = #J; --- this is an assertion
30    Confirm J = #I;
31
32    Write_Line(I);
33    Write_Line(J);
34  end Main;
35 end Activity_101_Reasoning_about_Assignments;
```

PROVED

VC 0_1 (29) ✓

VC 0_2 (30) ✓

Explicit Confirm Statement:
Activity_101_Reasoning_about_Assignments.fa(30)

Goal:
(I' = I')

Given:

1. (min_int <= I')
2. (I' <= max_int)
3. (min_int <= 0)
4. (1 <= max_int)

VC 0_3 (17) ✓

1) Clicking MP-Prove (Activity 1a) causes the Verifier to generate 3 VCs, one for Main's ensures at Line 17, and one for each Confirm, at Lines 29 & 30

2) The Verifier then proceeded to prove all 3 VCs. The green check mark means they proved

3) This Goal is like an RT Confirm These Givens are like RT Assumes

4) But the Verifier does not work in exactly the same way as an RT, so a Goal and Givens won't make sense to us