

# The Test Case Reasoning Assistant

Dana P. Leonard, Jason O. Hallstrom, Murali Sitaraman  
School of Computing  
Clemson University

This work is supported in part through grants from the National Science  
Foundation (DUE-0633506, CNS-0745846, DMS-0701187, CCF-0811748).



# Course Module

## Target

Any course where interface contracts can be introduced

## Objectives

- Reading and interpreting formal specifications
- Understanding abstract models of SW behavior
- Using components based on contract understanding

## Approach

Teach students to read, interpret, and apply interface specifications using test point creation exercises

# Pilot Studies

CpSc 215:  
Software Development  
Foundations

CpSc 372:  
Software Engineering  
Foundations

CS 349:  
Software Engineering

**CLEMSON**  
UNIVERSITY

**DENISON**  
UNIVERSITY

# Exercise #1

```
Function Mystery(preserves x : Integer) : Integer
requires
  x >= 0
ensures
  (result * result <= x) and
  ((result + 1) * (result + 1)) > x
```

## Question

What does the mystery function compute?

## Solution Strategy

Develop test cases that satisfy the specification

#x	result	check ?
0	0	$(0*0 \leq 0) \wedge (1*1 > 0)$
1	1	$(1*1 \leq 1) \wedge (2*2 > 1)$
4	3	$(3*3 \leq 4) \wedge (4*4 > 4)$
18	4	$(4*4 \leq 18) \wedge (5*5 > 18)$

# Exercise #2

```
Function Mystery(preserves x : Integer) : Boolean
requires
  x > 2
ensures
  Mystery() =
    there exists k, m : Integer
    s.t. (k >= 2) and (m >= 2) and (x = k * m)
```

#x	result	check ?
4	TRUE	$\exists k, m : (k \geq 2) \wedge (m \geq 2) \wedge (4 = k * m)$
7	FALSE	$\exists k, m : (k \geq 2) \wedge (m \geq 2) \wedge (7 = k * m)$
13	TRUE	$\exists k, m : (k \geq 2) \wedge (m \geq 2) \wedge (13 = k * m)$
22	TRUE	$\exists k, m : (k \geq 2) \wedge (m \geq 2) \wedge (22 = k * m)$

## Question

What does the mystery function compute?

# Abstract Models

Later, students are taught to reason about component behavior using abstract mathematical models

## Theory

**Str(Integer) : mathematical string of integers**

**examples :**  $\langle \rangle$ ,  $\langle 1 \rangle$ ,  $\langle 2, 1 \rangle$ ,  $\langle 3, 1, 2 \rangle$ ,  $\langle 4, 1, 2, 3 \rangle$

**concatenation :**  $\langle \rangle \circ \langle 1 \rangle = \langle 1 \rangle$ ,  $\langle 1, 2 \rangle \circ \langle 5, 6 \rangle = \langle 1, 2, 5, 6 \rangle$

**length :**  $|\langle \rangle| = 0$ ,  $|\langle 1, 3, 2 \rangle| = 3$

## Specification

**Stack is modeled by Str(Entry)**

**Operation Push(updates s : Stack, preserves x : Entry)**

**ensures**  $s = \langle x \rangle \circ \#s$

**Operation Pop(updates s : Stack, produces x : Entry)**

**ensures**  $\#s = \langle x \rangle \circ s$

**Operation Length(preserves s : Stack)**

**ensures result** =  $|s|$

# Exercise #3

**Operation** Mystery(**updates**  $s : \text{Stack}$ , **preserves**  $x : \text{Integer}$ )

**requires**

$$x \leq |\text{self}|$$

**ensures**

**there exists**  $l, r : \text{Str}(\text{Entry})$

**s.t.**  $(\#s = l \circ r)$  **and**  $(|l| = x)$  **and**  $(s = r \circ l)$

#s	#x	s	check ?
$\langle \rangle$	0	$\langle \rangle$	$\exists l, r : (\langle \rangle = l \circ r) \wedge ( l  = 0) \wedge (\langle \rangle = r \circ l)$
$\langle 1 \rangle$	1	$\langle 1 \rangle$	$\exists l, r : (\langle 1 \rangle = l \circ r) \wedge ( l  = 1) \wedge (\langle 1 \rangle = r \circ l)$
$\langle 1, 2, 3 \rangle$	2	$\langle 2, 3, 1 \rangle$	$\exists l, r : (\langle 1, 2, 3 \rangle = l \circ r) \wedge ( l  = 2) \wedge (\langle 2, 3, 1 \rangle = r \circ l)$
$\langle 1, 2, 3, 4, 5 \rangle$	3	$\langle 4, 5, 1, 2, 3 \rangle$	$\exists l, r : (\langle 1, 2, 3, 4, 5 \rangle = l \circ r) \wedge ( l  = 3) \wedge (\langle 4, 5, 1, 2, 3 \rangle = r \circ l)$

# The Reasoning Workbench

Smart classrooms enable new teaching tools and exercises that enhance the learning process:

- Excite and engage students
- Provide real-time performance feedback
- Monitor student learning and identify problem areas

## The Reasoning Workbench

A collection of reasoning assistants that guide students through reasoning exercises and provide real-time feedback as they work



# Test Case Reasoning Assistant



Test Case Reasoning Assistant  
(TCRA)

## Purpose

Assist students in developing test cases that measure and reinforce their understanding of abstract models and interface specifications

# TCRA Overview



Student Interface

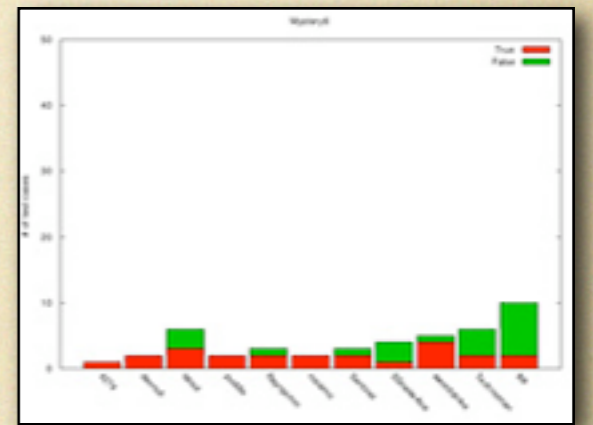
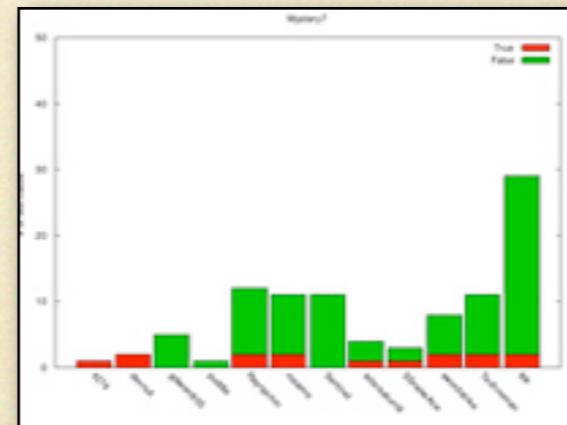
performance data

Instructor Interface



Exercise Repository

```
public class StackExercise implements TCRAExercise {  
    presentation checking  
}
```



individual / aggregate performance summaries

# TCRA Student Interface

Test Case Reasoning Assistant

File Tools

```

Concept Stack_Template(type Integer;
    evaluates Max_Depth: Integer);
uses Std_Integer_Fac, String_Theory;
requires Max_Depth > 0; (Max_Depth = 5)
Type Family Stack  $\sqsubseteq$  Str(Entry);
exemplar S;
constraints |S|  $\leq$  Max_Depth;
initialization ensures S =  $\wedge$ ;
    
```

- Mystery1
  - Mystery2
  - Mystery3

Argument Name	Value
#S	<1,2,3>
Stack #T	<4,5,6>
S	<2,3>
Stack T	<1,4,5,6>

OK

```

Operation Mystery3(updates S, T: Stack);
requires |S| > 0 and |T| > 0;
ensures
 $\exists E: \text{Integer} \ni \text{Rev}(S) * T =$ 
   $\text{Rev}(\#S) * \#T$  and
   $\langle E \rangle * S = \#S;$ 
    
```

```

Operation Mystery3(updates S, T: Stack);
requires |< 1, 2, 3 >| > 0 and |< 4, 5, 6 >| > 0;
ensures
 $\exists E: \text{Integer} \ni \text{Rev}(\langle 2, 3 \rangle) * \langle 1, 4, 5, 6 \rangle =$ 
   $\text{Rev}(\langle 1, 2, 3 \rangle) * \langle 4, 5, 6 \rangle$  and
   $\langle 1 \rangle * \langle 2, 3 \rangle = \langle 1, 2, 3 \rangle$ 
    
```

# TCRA Instructor Interface

The screenshot shows the TCRA Graph Generator application window. The title bar reads "TCRA Graph Generator". The menu bar includes "File", "Graph Options", and "Tools".

Under the "Tools" menu, there are three options: "On Campus Access" (selected with a radio button), "Off Campus Access" (unselected), and "Download Logs".

Below the menu is a "Choose Exercises." button. To the right, there are two lists:

- Loaded Files:** A list containing several files with names ending in "\_tcr\_log".
- Available Exercises:** A list containing "ResolveBoundedQueueExercise", "ResolveBoundedStackExercise" (highlighted), "StackReasoningExercise", and "ResolvePreemptableQueueExercise".

Below these lists is a table with two columns: "Exercise" and "Available Methods".

Exercise	Available Methods
ResolveBoundedStackExercise	Push
ResolveBoundedStackExercise	Pop
ResolveBoundedStackExercise	Depth
ResolveBoundedStackExercise	Rem_Capacity

Below the table is a text input field labeled "Enter Date Range (MM/DD/YYYY HH:MM)". It contains two date-time strings: "04/16/2008 00:00" and "04/16/2008 23:59".

Below the date range is a "Generate graph(s)." button.

At the bottom of the window, there are three bar charts. Each chart has a y-axis labeled "# of Exercises" and an x-axis with 10 categories. The bars are colored red and green. The first chart shows very low values. The second chart shows slightly higher values. The third chart shows significantly higher values, with the last bar reaching approximately 40.

# Additional Modules

More advanced modules require students to rely on specifications as part of their development work

## Contract Development in Teams

- ✎ Medium-scale project subdivided into components
- ✎ Students work independently using formal contracts
- ✎ Systems composed from selected implementations

## Tool-Assisted Program Verification

- ✎ Students derive simple verification conditions (VCs)
- ✎ Derivation process is reinforced using VC generator
- ✎ Students prove generated VCs using proof assistant

# Questions?

## The Test Case Reasoning Assistant

Dana P. Leonard, Jason O. Hallstrom, Murali Sitaraman

School of Computing

Clemson University

This work is supported in part through grants from the National Science Foundation (DUE-0633506, CNS-0745846, DMS-0701187, CCF-0811748).

