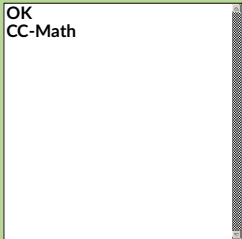


Compound Inequalities

(Also available for Pyret)

Students learn to compose inequalities using the concepts of union and intersection, and solve problems using compound inequalities. Finally, they apply what they've learned to set screen boundaries in their game.

Prerequisites	Simple Inequalities
Relevant Standards	Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere). 
Lesson Goals	Students will be able to: <ul style="list-style-type: none">Describe how functions can work together.Describe the solution set of a compound inequalityMake mathematical adjustments relevant to their game.
Student-Facing Lesson Goals	<ul style="list-style-type: none">I can use two or more inequalities together and describe the area they enclose.I can tell someone else how two or more <i>functions</i> work together.I can make adjustments to a program based on how the program behaves.
Materials	<ul style="list-style-type: none">Lesson slides template (Google Slides)Inequalities Explore worksheet (original (Page 45), solutions)Design Recipe: onscreen? (original (Page 46), solution) Bootstrap Formative Assessments <ul style="list-style-type: none">Booleans Review (Quizizz, Desmos Activity)
Preparation	<ul style="list-style-type: none">Make sure all materials have been gatheredDecide how students will be grouped in pairs
Supplemental Resources	<ul style="list-style-type: none">Desmos Inequalities Bundle (Desmos Activities)Inequalities & Graphing Inequalities (Quizizz)Inequality Graph Illustrator (Geogebra)Graphing Compound Inequalities (Quizizz)
Key Points for the Facilitator	<ul style="list-style-type: none">Role-playing can help students understand the job of <code>onscreen ?</code>, and how it relates to <code>safe-left?</code> and <code>safe-right?</code>.If a student's <code>TARGET</code> and <code>DANGER</code> image seem to be "getting stuck" on the edge of the screen, the student may have to adjust the side boundaries depending on the size of their images.The code for the boundary functions in the game is <i>exactly the same</i> as in Sam the Butterfly.

For a textbook-like version of materials similar to these, you may wish to see the [prior unit-based version](#)

Glossary

coordinate :: a number or set of numbers describing an object's location

function :: a mathematical object that consumes inputs and produces an output

Warmup

Students should have their computer, contracts page, and pencil and be logged in to [WeScheme](#) with their Game Project file open.

Compound Inequalities

10 minutes

Overview

Students consider the need to *compose* inequalities, and think about how to write them.

Launch

We use inequalities for lots of things:

- Is it hot out? (*temperature* $> 80^\circ$)
- Did I get paid enough for painting that fence? (*temperature* $< \$100$)
- Are the cookies finished baking? (*timer* $= 0$)

Have students come up with other examples.

But many times we need to *combine* inequalities:

- Should I go to the beach? (*temperature* $> 80^\circ$ and *weather* $=$ "sunny")
- Was this burrito worth the price? (*taste* $=$ "delicious" and *price* $\leq \$20$)

Have students come up with other examples.

Guide students through other examples of *and* and *or* with various statements, such as "I'm wearing a red shirt AND I'm a math teacher, true or false?" or "I'm an NBA basketball star OR I'm having pizza for lunch, true or false?". This can make for a good sit-down, stand-up activity, where students take turns saying compound boolean statements and everyone stands if that statement is true for them.

Investigate

Both mathematics and programming have ways of combining - or *composing* - inequalities.

Have students complete [Inequalities — Practice \(Page 45\)](#).

Synthesize

- Be really careful to check for understanding here. *Expressions using and only produce true if both of their sub-expressions are true*. *Expressions using or produce true if either of their sub-expressions are true*.

Strategies for English Language Learners

When describing compound inequalities, be careful not to use "english shortcuts". For example, we might say "I am holding a marker *and* an eraser" instead of "I am holding a marker *and* I am holding an eraser." These sentences mean the same thing, but the first one obscures the fact that "and" joins two complete phrases. For ELL/ESL students, this is unnecessarily adds to cognitive load!

Protecting Sam on Both Sides

30 minutes

Overview

Students solve a word problem involving compound inequalities, using `and` to compose the simpler boundary-checking functions from the previous lesson.

Launch

Note: In this programming language, question marks are pronounced "huh?". So `safe-left?` would be pronounced "safe left huh?" This can be a source of some amusement for students!

- Recruit three student volunteers to roleplay the functions `safe-left?`, `safe-right?` and `onscreen?`. Give them 1 minute to read the contract and code, as written in the program.
- As in the previous lesson, ask the volunteers what their name, Domain and Range are, and then test them out by calling out their name, followed by a number. (For example, "(safe-left? 20)!", "(safe-right? -100)!", "(onscreen? 829)!") **Note** the code for `onscreen` calls the *safe-left* function!. So the student roleplaying `onscreen` should turn to `safe-left` and give the input to them.

For example:

- **Facilitator:** "onscreen-huh 70"
- **onscreen?** (turns to safe-left?): "safe-left-huh 70"
- **safe-left?:** "true"
- **onscreen?** (turns back to facilitator): "true"

- **Facilitator:** "onscreen-huh -100"
- **onscreen?** (turns to safe-left?): "safe-left-huh -100"
- **safe-left?:** "false"
- **onscreen?** (turns back to facilitator): "false"

- **Facilitator:** "onscreen-huh 900"
- **onscreen?** (turns to safe-left?): "safe-left-huh 900"
- **safe-left?:** "true"
- **onscreen?** (turns back to facilitator): "true"

Ask the rest of the class

- What is the problem with `onscreen?` ?
It's only talking to `safe-left?`, it's not checking with `safe-right?`
- How can `onscreen?` check with both?
It needs to talk to `safe-left?` AND `safe-right?`

Have students complete **Word Problem: onscreen?** (Page 46). When this functions is entered into WeScheme, students should now see that Sam is protected on `_both` sides of the screen.

Extension Option

What if we wanted to keep Sam safe on the top and bottom edges of the screen as well? What additional functions would we need? What functions would need to change?}

Overview

Students identify common patterns between 2-dimensional boundary detection and detecting whether a player is onscreen. They apply the same problem-solving and narrow mathematical concept from the previous lesson to a more general problem.

Launch

Have students open their in-progress game file and press Run.

- How are the `TARGET` and `DANGER` behaving right now?
They move across the screen.
- What do we want to change?
We want them to come back after they leave one side of the screen.
- How do we know when an image has moved off the screen?
We can see it.
- How can we make the computer understand when an image has moved off the screen?
*We can teach the computer to compare the image's **coordinates** to a numeric boundary, just like we did with Sam the Butterfly!*

Investigate

Students apply what they learned from Sam the Butterfly to fix the `safe-left?`, `safe-right?`, and `onscreen?` functions in their own code.

Since the screen dimensions for their game are 640x480, just like Sam, they can use their code from Sam as a starting point.

Common Misconceptions

- Students will need to test their code with their images to see if the boundaries are correct for them. Students with large images may need to use slightly wider boundaries, or vice versa for small images. In some cases, students may have to go back and rescale their images if they are too large or too small for the game.
- Students may be surprised that the same code that "traps Sam" also "resets the `DANGER` and `TARGET`". It's critical to explain that these functions do *neither* of those things! All they do is test if a coordinate is within a certain range on the x-axis. There is other code (hidden in the teachpack) that determines *what to do if the coordinate is offscreen*. The ability to re-use function is one of the most powerful features of mathematics - and programming!

Additional Exercises:

- Word Problem: hot?
- Word Problem: sunny?
- Word Problem: beach-day?