Asa Oines
6.890 Final Project Write-up
12/2014

Visualizations of Block Pushing Hardness Reductions

Introduction

For me, one of the most compelling aspects of 6.890 has been looking at video games from a hardness perspective. Prior to 6.890 I had never considered that games I play for fun would be worth examining through this lens. This was the motivation for my final project, for which I created a playable visualization of the 3SAT to Push-* reduction. Hopefully this tool will be used in future presentations of this proof to help illustrate the mechanics of the reduction.

Push-* and Block Pushing

Push-* is one of a number of block pushing puzzles discussed in class. In this class of problems a robot moves around a rectangular grid. Each square on this grid can be empty or contain a block. The robot has strength *n*, and can push *n* blocks at a time so long as there are empty spaces to push the blocks into. If the robot is able to reach the goal square then the problem is solved. In the Push-* problem the robot has infinite strength and is thus able to push infinitely many blocks at a time.

3SAT to Push-*

Push-* can be proven NP-Hard by a reduction from 3SAT. This reduction makes use of four gadgets; a variable gadget, bridge gadget, clause gadget, and connection gadget. In each variable gadget the robot must perform actions that set one or both literals to false. Initially there exist holes in the connection gadget for each literal in each clause. To advance

to the next variable gadget the robot must push blocks to the right, filling holes in the connection gadget. As the robot navigates the variable gadgets the holes corresponding to the literals that have been set to false are filled.

After the robot has completed moving through the variable gadgets it will reach the bridge gadget. The bridge gadget provides a unidirectional path for the robot to travel from the variable gadgets to the clause gadgets. This ensures that after the variables have been set the robot can not return to the variable gadgets. For each clause gadget, if one of the literals in the clause has been set to true a hole exists in the connection gadget below that clause gadget. The robot can then traverse the clause gadget by pushing a block down into the available hole and moving right to the next clause. If the robot is able to traverse each of the clause gadgets it is able to reach the goal square and thus win the game.

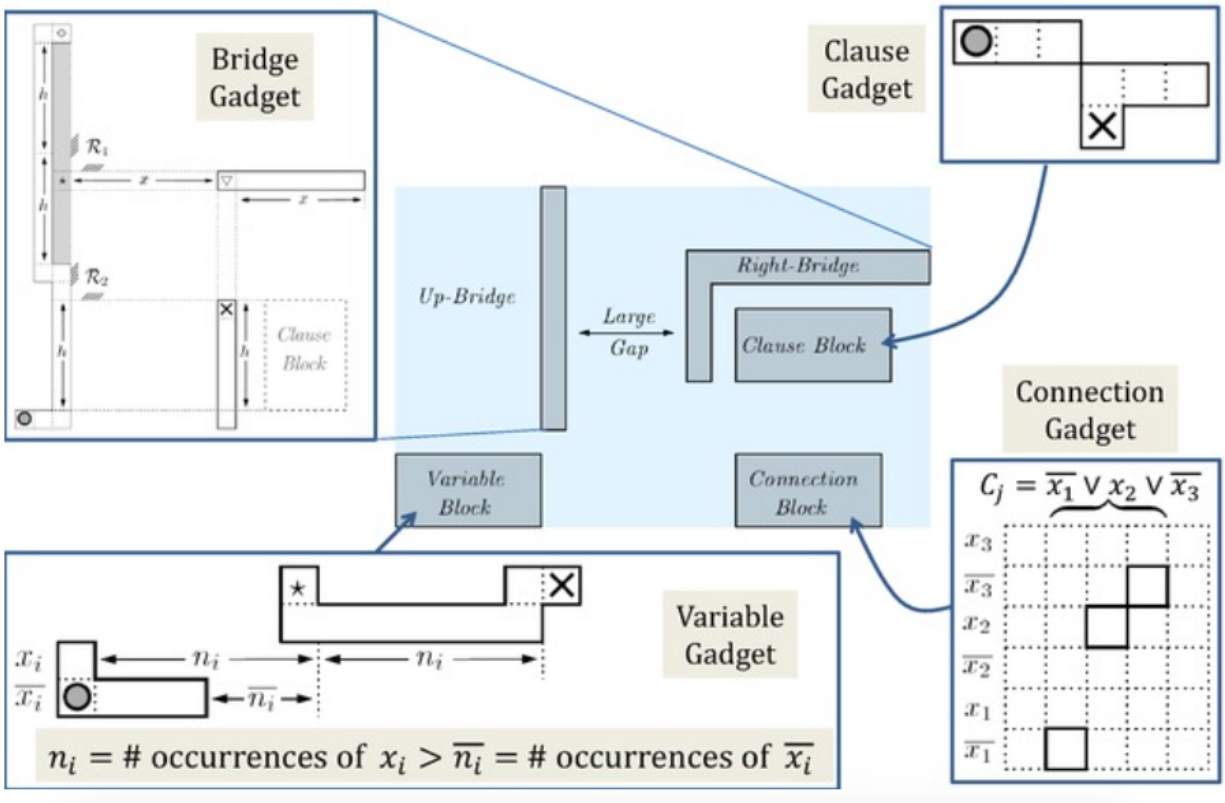

Figure 1: Gadgets used in the 3SAT to Push-* reduction

Visualizing the 3SAT to Push-* Reduction

The goal of visualizing the 3SAT to Push-* reduction is to make the proof more easily understood by the user. To do this effectively it was important to think carefully about what features should be included in the visualization to make the reduction as clear as possible. After careful consideration I decided that the visualization had to have the following properties.

Playable

The feature that I deemed most important was that the visualization be playable. By allowing the user to interact directly with the Push-* instance the user can explore for themselves how their actions enable or prevent them from reaching the goal square. Direct interaction with the proof is the best way to show the user how each gadget works.

Gadgets can easily be traced back to their roots in the 3SAT instance

To really understand the proof it is important to understand where the different parts of the Push-* instance come from. By clearly showing the connection between corresponding elements in the 3SAT and Push-* instances the logic behind the construction of the Push-* instance becomes more plain.

Actions in Push-* are reflected in 3SAT

When playing the Push-* instance the user performs actions that have analogs in 3SAT. For example, by navigating the variable gadgets in Push-* the user is setting literals to false in 3SAT. By traversing the clause gadgets the user is verifying that the clauses are true. Just as it is important for the user to be able to see how the Push-* instance relates to the

3SAT instance from which it originates, so too is it important for the user to be able to see how actions in Push-* correspond to actions in 3SAT.

Actions in 3SAT are reflected in Push-*

For the user to really understand the reduction showing how actions relate in the opposite direction is just as important. Most users should be more familiar with 3SAT and demonstrating how a specific set of variable assignments will lead to a unique sequence of actions in Push-* will help clarify the link between the two problems. We can expect many of the users to be unfamiliar with this reduction and they likely will not understand what actions to take to set the variables as they wish. By allowing the user to see the mapping of variable assignments to robot movements the Push-* instance becomes much more accessible to first time users.

Implementation

To make my project easily accessible on the web I chose to write my visualization in Javascript. The Push-* instance is rendered using the HTML5 canvas element which comes with several useful methods for displaying the board including rectangle drawing, scaling, and translation.

To interact with the visualization the user must first create a 3SAT instance. Initially the page displays three input fields and an “Add Clause” button that can be used to build a 3SAT formula. After adding the first clause the user has the option to continue to build a formula by adding and removing clauses in the same manner. The 3SAT instance is stored in the URL hash as a two dimensional JSON encoded array, which allows users to share 3SAT formulas and their corresponding Push-* instances by sharing URLs.

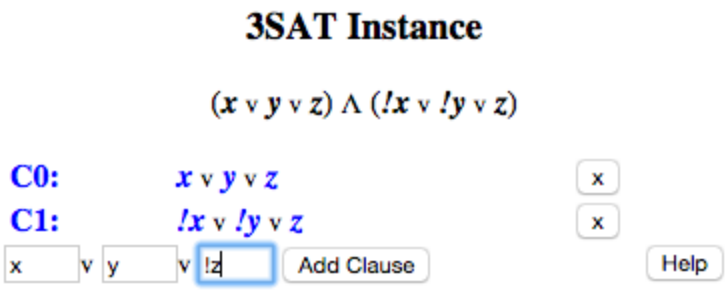

Figure 2: UI for creating a 3SAT formula

Once the user has a non-empty 3SAT formula the Push-* instance is created. The user can navigate the Push-* instance using the “w”, “a”, “s”, and “d” keys. Additionally the “u” key will undo the user’s most recent move, the “r” key will reset the game board to it’s initial state, and the “z” key will toggle between a zoomed in view centered on the current position of the robot and a zoomed out view showing the board in it’s entirety. The user can see these controls by clicking on the “Help” button.

The game board is stored in a two dimensional array of ones and zeros. Ones represent a square with a block in it while zeros represent empty space. Whenever the 3SAT formula is changed the Push-* instance is regenerated. The program calculates the expected number of rows and columns needed to represent that instance and creates a two dimensional array of ones of that size. The variable gadgets, bridge gadget, clause gadget, and connection gadgets are then computed and placed on the board in their appropriate positions. As the robot pushes blocks the two dimensional array representation of the board is modified accordingly.

To show how the Push-* instance is constructed from 3SAT, the rows in the variable gadgets are labeled with their corresponding literals. Similarly the columns in the clause and connection gadgets are labeled with their corresponding clauses and literals.

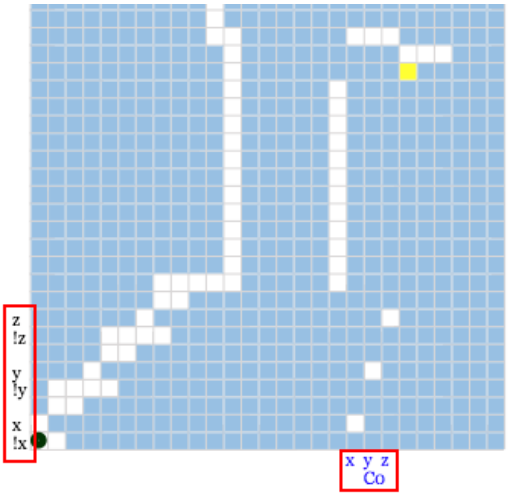

Figure 3: Labeled variable, clause, and connection gadgets for the formula (***x*** v ***y*** v ***z***)

To navigate Push-* the user must perform actions that are analogous to assigning values to the boolean variables in 3SAT. The visualization clues the user in to these actions by color coding the literals in the 3SAT formula by the value set by the user in Push-*. In the initial state of the game all of the literals are set to true. As the user plays the game he or she is forced to set some of these literals to false. When this occurs the literals in the 3SAT formula are changed from blue to red to indicate that the user has set that literal to false. Likewise when all of the literals in a clause have been set to false the label for the clause is changed from blue to red.

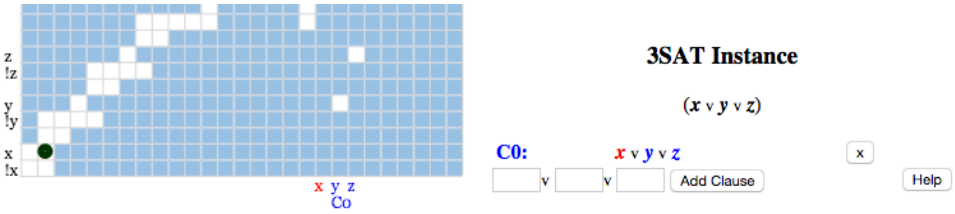

Figure 4: Board and formula after setting “x” to false

Many users may not be familiar with the reduction and will not know how to navigate the Push-* instance successfully. To assist with this the visualization has a “Hint” feature that allows users to view the mapping between variable assignments and robot moves. Users can

enter in their variable assignments and the hint feature draws on the board the sequence of moves that maps to this assignment. This makes the proof accessible to users who are familiar with 3SAT but unfamiliar with Push-* and the reduction being visualized.

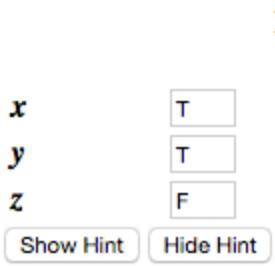

Figure 5: UI for generating a hint for the 3SAT formula (***x*** v ***y*** v ***z***)

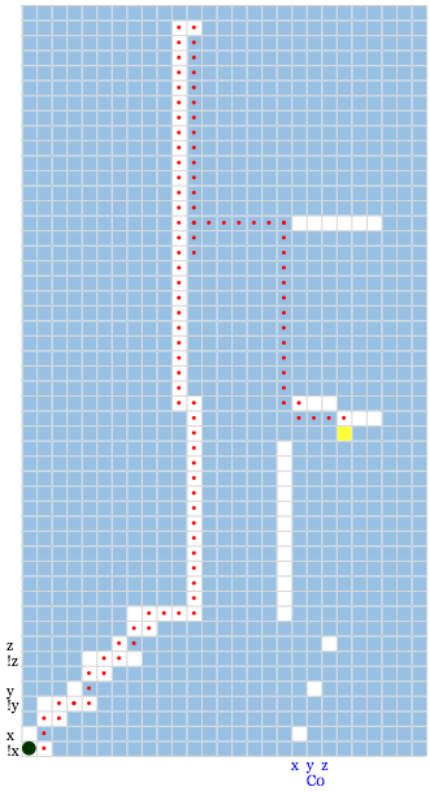

Figure 6: The hint generated from figure 5. Following the red dots will set ***x*** and ***y*** to true and ***z*** to false

Conclusion

A proper visualization of a reduction makes clear the connections between the problem being reduced from and the problem being reduced to. By illustrating the

connections between the instances of 3SAT and Push-* and the actions in one problem to the actions in the other, the visualization I created will be an effective teaching tool for users who are unfamiliar with the reduction. This tool can be used in future demonstrations of this reduction as well as provide a model for other block pushing visualizations.