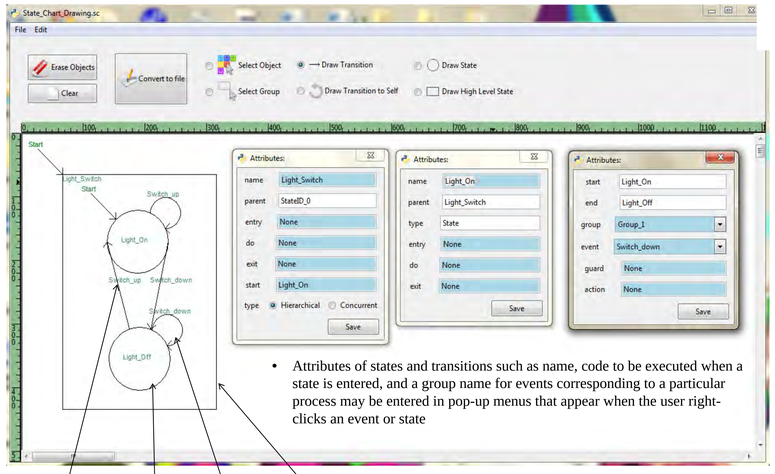
Ionospheric Radar Experiment Scenario Modeling

Natalie Larson, Vanderbilt University
MIT Haystack Observatory REU Summer 2011
Mentors: Bob Schaefer, Phil Erickson

ABSTRACT

MIT Haystack's Atmospheric Sciences Group has for nearly 50 years operated the Millstone Hill upper atmospheric radar, focused on studies of the near-Earth space environment. The radar is a complex system, which can pose a challenge for experiment design. The aim of this project was to build a graphical user interface to allow a scientist to easily design and model scenarios of experiment flow through the Millstone Hill radar experiment chain. This end was achieved by creating an interface that allows a user to draw a state chart and convert the drawing into a parameter file that can then be converted into python code for controlling the radar. The software supports such functions as moving objects, cutting, pasting, saving and opening files, undoing actions, redoing actions, adding and editing state and transition attributes, and choosing attributes from previously entered data. The software also performs a number of tasks to automatically ensure that a drawing is consistent. While open source software exists to translate code into state chart drawings, no low-cost readily available software could be found to translate state chart drawings into code. For this reason, and for the flexibility and control self-made software affords, the state chart drawing program was created from scratch.

PROGRAM USE



- Attributes of states and transitions such as name, code to be executed when a state is entered, and a group name for events corresponding to a particular process may be entered in pop-up menus that appear when the user right-clicks an event or state

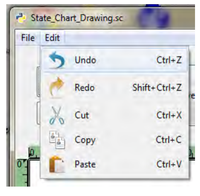
State

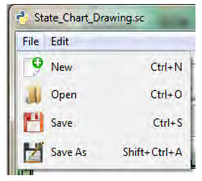
Transition to self

Super-state: Composed of other states
-May be either hierarchical or concurrent

Transition: Event that causes one state to transition to another

HIERARCHICAL	CONCURRENT
States contained must be visited serially	States contained may be visited independently
Processed on one computer	Processed in parallel on multiple computers
States contained must have one and only one start transition	States contained must each have a start transition

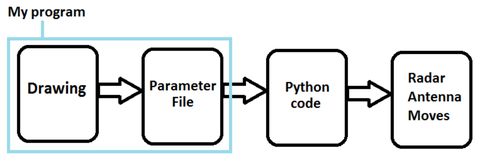




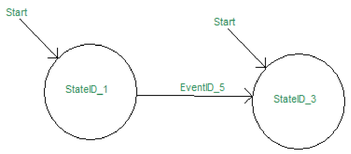
Updates made automatically by the program:
-Number of start transitions within a super-state
-Parents of objects
-Start and end fields for transitions
-Snap-to-fit arrows

Available Menu Bar items

PROGRAM FLOW



SAMPLE DRAWING



-State 1 is a required, implicit concurrent base and transition 1 is a required, implicit start transition for State 1

-States must be listed with parent states appearing before child states

SAMPLE PARAMETER FILE

[STATE_1]
name = StateID_0
parent = None
type = ConcurrentState
entry = None
do = None
exit = None

[STATE_2]
name = StateID_1
parent = StateID_0
type = State
entry = None
do = None
exit = None

[STATE_3]
name = StateID_3
parent = StateID_0
type = State
entry = None
do = None
exit = None

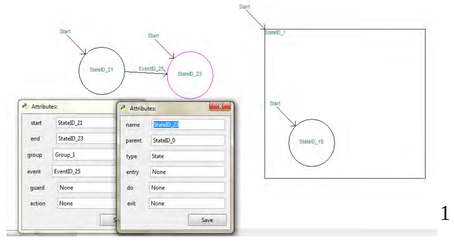
[TRANSITION_1]
start = start
end = StateID_0
event = None
guard = None
action = None

[TRANSITION_2]
start = start
end = StateID_1
event = None
guard = None
action = None

[TRANSITION_3]
start = start
end = StateID_3
event = None
guard = None
action = None

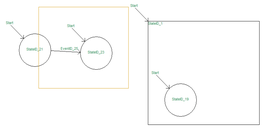
[TRANSITION_4]
start = StateID_1
end = StateID_3
event = Group_1.EventID_5
guard = None
action = None

USE CASE: MOVING AN OBJECT



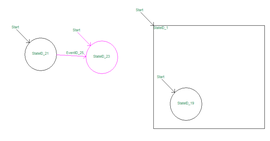
1

1. The parent fields of states and the start and end fields of transitions are sensed and automatically recorded upon any creation, move, or paste of a state or event. Note the parent of StateID_23 and start and end attributes of EventID_25 before the move.



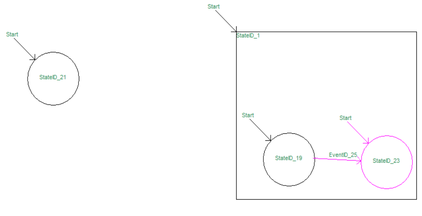
2

2. Select Groups of objects using the Select Group button, which creates a rectangular drag tool. Single objects may be selected with the "Select Object" button, by clicking within a delta distance of an object's perimeter.



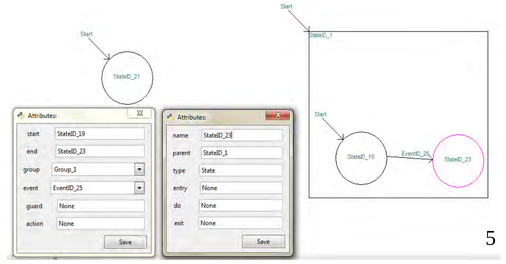
3

3. Selected objects are highlighted and can then be dragged.



4

4. The mouse button has not yet been released. The arrow representing EventID_25 overlaps the circle representing StateID_19.



5

5. The mouse button has been released and the program has automatically corrected the length and attachment point of EventID_25, has deleted the start arrow pointing to StateID_23 since its super-state, StateID_1, is Hierarchical, and has updated the parent attribute of StateID_23 and the start and end attributes of Event ID_25.

FEATURE	ALGORITHM
Saving and opening files	Pickling: Python function that turns objects into byte streams
Keeping start arrows consistent	Find the smallest surrounding rectangle to find parent type: hierarchical or concurrent, and update arrows according to type. Check for consistency every time an object is created, moved, pasted or erased. If parent is concurrent: check that each child object has one start arrow and make changes as necessary. If hierarchical: look for a start arrow within the rectangle, keep the first arrow found and erase all other start arrows, else if no start arrow was found, assign a start arrow to the first object found .
Real-time drawing of arrows	Change coordinates of arrowhead to current cursor location and continually redraw the screen
Real-time drawing of rectangles and ability to drag a rectangle in any direction	Calculate the vector that starts at the start click and ends at the current mouse location; rotate it 45 degrees in each direction to form the rectangle. The uppermost, leftmost, point on the rectangle and its width and height are the input to a GTK rectangle drawing function.
Snap-to-fit arrows	If the point clicked is within a delta distance of a certain shape's perimeter, create a vector from the shape's center to the point clicked; if the object is a circle, scale the vector by the radius of the circle to obtain the snap-to point on the circle; if the object is a rectangle, determine which edge will be crossed by the vector and find the intersection of the vector and the line segment of the rectangle's edge to find the snap-to point.
Highlighting/Selecting objects	Is the click within a delta distance of any object's perimeter? Arrow: does the point lie within the rectangle created by delta d on either side and on either end of the line segment? Circle: does the point satisfy the equation of the circle, accounting for delta d? Rectangle: does the point lie within any of the four rectangles created by delta d on either end and on either side of each line segment?
Ensuring consistency of parent fields of states and start and end fields of transitions	Basis: find the smallest surrounding rectangle to find an object's parent. Currently the code accounts for a number of specific conditions individually. For example, if a rectangle is moved or erased, make the parent field of all objects that previously had that rectangle as a parent the moved rectangle's former parent. This approach does not waste time or space making unnecessary updates but is more complex than an approach which checks all objects with every move, create, paste, and erase action.
Undo/Redo	Keep a set of super-lists to hold the current values of all variables needed to recreate the current drawing. Append new values to each of the lists any time an action is taken. Undo an action by moving backward through the set of lists; redo by moving forward.
Moving object(s)	For all highlighted objects, draw vectors from the original point clicked to all object parameters needed to draw the object (for example, radius, for a circle). As the cursor moves, update the object's parameters to be the current location of the mouse minus the vector corresponding to the parameter.
Printing states so that each parent state is printed before its child	Use a depth-first-search: add the base state to a list. While the list is not empty, pop the last state from the list, write that state's attributes to the file, and add its children to the list. Print all circles afterward.

FUTURE WORK

- Print and print preview buttons
- Capability to represent events that are turned into messages and then back into events to facilitate distributed computing
- Ability to draw loops at any location on a shape
- Rotation of event names so that they appear at the same angles as the arrows to which they correspond
- Ability to extend or contract lengths of arrows when objects or groups of objects are moved, if keeping all objects attached is desired
- Possible standardization of all shapes as rectangles
- Way to show movement that does not cause the screen to blink